MIXED INTEGER PROGRAMMING

DISSERTATION

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE OF

Master of Philosophy

IN

OPERATIONS RESEARCH

BY

AHMAD YUSUF ADHAMI

Under the Supervision of

DR. ABDUL BARI

DEPARTMENT OF STATISTICS & OPERATIONS RESEARCH
ALIGARH MUSLIM UNIVERSITY
ALIGARH (INDIA)

2005

Phone : (0571) 401251

DEPARTMENT OF
STATISTICS & OPERATIONS
RESEARCH
ALIGARH MUSLIM UNIVERSITY
ALIGARH – 202 002, INDIA

CERTIFICATE

This is to certify that Mr. Ahmad Yusuf Adhami has carried out the work reported in the present dissertation entitled "Mixed Integer Programming" under my supervision. The dissertation is Ahmad's own work and I recommend it for consideration for the award of Master of Philosophy in Operations Research.

(Dr. Abdul Bari)
Supervisor

ACKNOWLEDGEMENT

I hereby, with deep sense of gratitude, wish to express my sincere regards and thanks to my supervisor **Dr. Abdul Bari**, for his excellent supervision, invaluable advice, and encouragement during the completion of this work.

My thanks are also due to **Prof. M.Z. Khan**, Chairman, Department of Statistics and Operations Research, Aligarh Muslim University, Aligarh, for providing me the required facilities in the department.

I would like to acknowledge my indebtedness to a number of individuals for their assistance. Specially, to my seniors Mr. Quazzafi, Mr. Zaki, Mr. Maroof, Mr. Abrar and Mr. Bakshi, who were always with me with their helping hands.

My friends, Faisal, Israr, Shiraz and Danish bh., all deserves my sincere thanks for their cooperation.

Finally, I would like to express my deep sense of respect and thanks to my beloved parents, brother and sister, for their constant encouragement and Dua, which worked as a great expediter throughout my work.

(Ahmad Yusuf Adhami)

CONTENTS

PREFACE

The present dissertation is devoted to the study of "Mixed Integer Programming". It consists of five chapters with comprehensive list of references at the end. Each chapter is structured so that the introductory and background material are presented first. Symbols representing vectors or matrices are set in boldface type.

Chapter I contains an introduction to the field. It presents a short introductory discussion on mathematical programming, linear and integer programming. Further, the formulation approach, and brief introduction to solution methods are some of the subjects that have been covered in this chapter.

Chapter II examines the various methods for solving the integer programming problem. Particularly, cutting plane methods, and branch-and-bound methods are discussed in detail.

Chapter III presents the techniques for solving mixed integer programs. Mixed zero-one integer program is also treated in this chapter.

Chapter IV deals with the solution of nonlinear mixed integer programming problem. It gives several algorithms for solving mixed integer nonlinear optimization problem.

The closing chapter (chapter V) discusses the method of solving mixed integer programming by partitioning.

CHAPTER I

INTRODUCTION

Since the advent of the industrial revolution the world has seen a remarkable growth in the size and complexity of organizations. The changes in the structure of human organizations, specialization in various fields and introduction of division of labour concept in each organization made these more complex However, along its blessings, this increasing specialization has created new problems .One problem is a tendency for the many components of an organization to grow into relatively autonomous empires with their own goals and value systems, thereby losing sight of how their activities and objectives mesh with those of the overall organization. There is always a possibility that any policy, which is best for a particular component, may be detrimental for other components. All this leads the executive head of the organization to perform the most difficult duty of allocating the available resources to the various components and to coordinate the policies of different components in such a way that it serves the best interest of the organization as a whole. A wrong decision made at any stage can be of tremendous loss to the organization.

Another important development in modern times is that in competitive world one has to take a quick decision because any delay or postponement in it may give advantage to the competitive organization. Here by a decision we mean recommendation for the choice of a particular course of action from number of alternative courses, which is most useful for the organization. From above, it is clear that in modern times due to conflicting interests and competitive strategies the art of decision-making has become very complicated and important. A sound and useful decision requires vigorous and scientific approach to the problem. The application of Operations Research (O.R.) methods helps in making decisions in

such complicated situations. These kinds of problems and the need to find a better way to solve them provided the environment for emergence of O.R.

1.1 MATHEMATICAL PROGRAMMING PROBLEMS:

Mathematical programming (M.P.) is one of the many facets of the field of Operations Research. Mathematical programming can best be defined as a techniques used to find the optimum relationship between a number of inter-dependent variables or a means of obtaining the very best course of action where many courses of action exist. Mathematical programming shares the same type of problem-solving approach as any other type of operations research techniques. Mathematical programming provides a quantitative basis for management decisions. A basis with which management manipulates and controls various activities to achieve the optimal outcomes of managerial problems. Management can be more effective and efficient in making judgements by the use of mathematical programming. Mathematical programming can provide management with information about resource allocation. In mathematical programming analyses, decisions are made under certainty or near certainty. That is, information on available resources and the relationship between variables are known. Therefore decisions models will help choose actions, which will invariably lead to optimal or nearly optimal results. Thus, mathematical programming can assist management in the selection of the most effective and desirable course of action from a number of available alternatives. Management problems are formulated in the form of mathematical models, which describe the quantitative features of all types of industrial problems.

Generally stated, the problem in M.P. is to find the unknown values of some variables, which will optimize the value of an objective function subject to a set of constraints. A general mathematical programming problem can be stated as follows:

Maximize (or Minimize) $z = f(x)$

Subject to $g_i(\mathbf{x}) \leq or = or \geq b_i$ (for i=1,2...n)

and $\mathbf{x} \geq \mathbf{0}$

Where z = value of the objective function which measures the effectiveness of the decision choice

$g_i(\mathbf{x})$ = set of i^{th} constraints.

$\mathbf{x}$ = unknown variables that are subject to the control of the decision maker

b_i = available productive resources in limited supply.

The objective function is a mathematical equation describing a functional relationship between various decision variables and the outcome of the decisions. The objective function may be either a linear or non-linear function of variables. The objective of the decision maker is to select the values of the variables so as to optimize the value of the objective function, z. The unknown variables whose values are to be chosen are called the decision variables in mathematical programming. The production quantity, sales price, number of days of plant operations, units of a product shipped to different destinations is only a few of the many examples of decision variables. The decision variables may take on fractional or integer values. Also, they may be discrete or continuous, depending on the business problem being analyzed. The limited supply of the productive resources imposes the constraints in mathematical programming. Mathematical programming problems do not exist unless the supply of some type of productive resources is limited. These restricted resources are expressed as equalities or inequalities in mathematical programming models. The decision maker's goal is to

find the values of the decision variables within limits, to optimize the value of the objective function.

Mathematical programming has been successfully applied in the following areas:

(i) Product Allocation: With a number of jobs that can run on a number of different machines, it is possible to determine how to best allocate the work to the machines so as to minimize either the total time or total cost to produce the entire work load.

(ii) Distribution and shipping: With a product demand at various locations and a supply of product at several warehouses, it is possible to determine which warehouse should ship how much product to which customer so that the total distribution costs are a minimum.

(iii) Job and Salary evaluation: Here mathematical programming is used in place of multiple correlation analysis to determine the relative weights of the factors considered. This applies to salary and executive type jobs. A similar analysis can be applied to any testing situation to give a better over-all evaluation.

(iv) Production Planning: It is possible to develop the lowest-cost producing plan, starting with a sales forecast, available plant capacity, and the tangible cost factors.

1.2 LINEAR PROGRAMMING PROBLEMS:

Linear programming is the part of mathematical programming that studies optimization problems having objective function and constraints of particularly simple form. It deals with that class of programming problems for which all relations among the variables are linear both in constraints and the

function to be optimized. The linear programming problem can be defined as the problem of finding non-negative (≥ 0) values of a given set of variables which minimize (or maximize) a given linear function in these variables under certain restrictions specified by linear equations or inequations.

The mathematical model of a LPP may be

$$\text{Minimize (or Maximize)} \quad c_1x_1 + c_2x_2 + \ldots + c_nx_n$$

$$\text{Subject to} \quad a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \geq b_1$$

$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \geq b_2$$

$$\vdots \qquad\qquad \vdots$$

$$\vdots \qquad\qquad \vdots$$

$$a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \geq b_n$$

$$x_1, x_2, \ldots x_n \geq 0$$

Here $c_1x_1 + c_2x_2 + \ldots\ldots + c_nx_n$, is the objective function to be minimized (or maximized) and will be denoted by z. The coefficients $c_1, c_2, \ldots, c_n$ are the (known) cost coefficients and $x_1, x_2, \ldots, x_n$ are the decision variables to be determined. The inequality $\sum_{j=1}^{n} a_{ij}x_j \geq b_i$ denotes the i^{th} constraint (or restriction). The column vector whose i^{th} component is b_i, which is referred to as the right hand side vector, represents the minimal requirements to be satisfied. The constraints $x_1, x_2, \ldots, x_n \geq 0$ are the non-negativity constraints. A set of variables $x_1, x_2, \ldots, x_n$ satisfying all the constraints is called a feasible point or a feasible

vector. The set of all such points constitutes the feasible region or the feasible space.

1.3 INTEGER PROGRAMMING PROBLEMS:

Any decision problem (with an objective to be maximized or minimized) in which the decision variables must assume nonfractional or discrete values may be classified as an integer optimization problem. In general, an integer problem may be constrained or unconstrained and the functions representing the objective and constraints may be linear or nonlinear. An integer problem is classified as linear if, by relaxing the integer restriction on the variables, the resulting functions are strictly linear.

An integer solution to a given linear programming problem can be obtained by rounding off the fractional values derived in the optimal solution by the simplex method. However if a feasible solution is obtained by rounding, one should not be under the illusion that such a solution is optimal or even close to optimal. The rounding procedure at best may be regarded as a heuristic. But even in this case, there may be other heuristics that would yield better solutions than when rounding is used. Any integer model having an original inequality constraint can never yield a feasible integer solution through rounding. This is based on the assumption that only basic variables can be rounded, if necessary, and that all the non-basic variables remains at zero level. This assumption is not unreasonable since it is generally difficult to consider elevating a non-basic variable above zero while maintaining feasibility. This was observed by Glover and Sommer(1972) [27]. To alleviate these shortcomings, a special solution procedure referred to as integer programming was developed.

1.4 Mathematical Definition of the Integer Programming:

The general integer program may be defined as

$$\text{Maximize (or minimize) } z = g_0(x_1, x_2, \ldots, x_n) \qquad \ldots\ldots(1)$$

$$\text{Subject to } g_i(x_1, x_2, \ldots, x_n) \leq or = or \geq b_i, \quad i \in M \equiv \{1,2,\ldots,m\} \qquad \ldots\ldots(2)$$

$$x_j \geq 0, \qquad j \in N \equiv \{1,2,\ldots n\} \qquad \ldots\ldots(3)$$

$$x_j \text{ an integer}, \quad j \in I \subseteq N$$

If I = N, that is, all the variables x_j are restricted to integer values, the problem is called a pure integer program. Otherwise, if $I \subset N$, then one is dealing with mixed program. Equation (1) is the objective function, which can be a linear or non-linear function. Expression (2) and (3) are the constraints and non-negative restrictions respectively. In the absence of the integrality condition, the problem becomes an ordinary (continuous) linear or non- linear program.

Applications of IP problems:

1. The Fixed Charge Problem:

The fixed charge problem is characterized by "one shot" outlays (or setup costs) that are incurred in the process of starting or renewing a business venture. For example, a manager who is faced with deciding which of several machines to buy, automobile plants to build, oil wells to drill, must account not only for the

continuing costs of operation but also for the initial fixed cost required to initiate the projects.

A typical fixed charge problem has the form:

$$\text{Minimize} \quad \mathbf{cx} + \alpha \mathbf{y}$$

$$\text{Subject to} \quad \mathbf{Ax} \leq \mathbf{b}$$

$$\mathbf{x} \geq \mathbf{0}, \mathbf{y} \geq \mathbf{0}$$

where the vectors **x** and **y** have the same dimension and $x_j > 0$ implies $y_j = 1$.The stipulation that $x_j > 0$ implies $y_j = 1$ conveys the meaning that to make, buy or, process any positive amount of x_j incurs a fixed charge of α_j (>0). However, the preceding formulation is not acceptable for IP since the constraints " $x_j > 0$ implies $y_j = 1$" is "logical" rather than linear. To put the problem in acceptable form, we assume the existence of a bound U_j so that $x_j \leq U_j$ is satisfied for all values of x_j compatible with $\mathbf{Ax} \leq \mathbf{b}$, $\mathbf{x} \geq 0$. Then the constraint $U_j y_j \geq x_j$ always holds for $y_j = 1$, and moreover, if y_j is integer valued, then $x_j > 0$ implies $y_j \geq 1$ and hence $y_j = 1$ in minimizing solution.

Thus the IP formulation of the fixed charge problem may be written as

$$\text{Minimize} \quad \mathbf{cx} + \alpha \mathbf{y}$$

$$\text{Subject to} \quad \mathbf{Ax} \leq \mathbf{b}$$

$$\mathbf{x} - \mathbf{Uy} \leq \mathbf{0}$$

$$\mathbf{x} \geq \mathbf{0}, \mathbf{y} \geq \mathbf{0}, \ \mathbf{y} \text{ is integer}$$

where $\mathbf{U} = (U_j)$.

2. Single hub air cargo fleet planning [35] (An MILP):

Here we present an MILP model, useful in maximizing profit, when the service network of an all cargo airline consists of a single spider graph i.e. a routine network in which only one special node (or hub or junction) may have more than two spider legs (A spider leg is an arc that connects two nodes). The scope of the model is confined to one typical night's operation. The following explanations are given for the understanding of the model.

The structure of the spider graph is specified by listing, for each spider leg (s=1,2,...,S), the cities that lie on that leg {city(s,1),...,city(s,m_s)}, in the returning order from the hub. It is also understood that no city may lie on more than one leg and each individual airplane will make one round trip on a single spider leg. For each leg

s= 1,2,...,S we are given the set: $M_{(s)}$ = {K | aircraft type K is available for leg S}. The distances of the airports are measured in the units of great circle miles.

The remaining data are: D_{ij} = the amount that customers want to send from city i to city j (1000 KG's).

c_{2ij} = the revenue received for carrying cargo from city i to city j (Rs/1000 KG's)

h(k) = the cost of one take off and landing from an aircraft of type K (Rs/great circle mile)

$t_{(k)}$ = the operating cost for an aircraft of type K(Rs/great circle mile)

$\beta_{(k)}$ = the fuel burn rate for an aircraft of type K(gallons/great circle mile)

f = the total amount of fuel available (gallons)

Δ_{ij} = distance from i to j

C_{1sk} = the cost of serving spider legs with an aircraft of type K

$$=2(h_{(k)}+t_{(k)}\Delta_{(hub,city(s,1))})+2\sum_{t=1}^{m_s-1}(h_{(k)}+t_{(k)}.\Delta_{(city(s,t),city(s,t+1))})$$

a_{sk} = the amount of fuel consumed

$$=2\beta_{(k)}.\Delta_{(hub,city(s,1))}+2\sum_{t=1}^{m_s-1}\beta_{(k)}.\Delta_{(city(s,t),city(s,t+1))}$$

where in both these formulae (comprising c_{1sk} and a_{sk}) the summation appears only if $m_s \geq 2$.

The MILP model to the problem is given by:

$$\text{Maximize } z = -\sum_{s}\sum_{k\in M_{(s)}} c_{1sk}x_{sk}+\sum_{i}\sum_{j\neq i} c_{2ij}y_{ij}$$

$$\text{Subject to } \sum\sum a_{sk}x_{sk} \leq f \qquad \text{.....(1)}$$

$$0 \leq y_{ij} \leq D_{ij}$$

$$x_{sk} = (0,1)$$

The decision variables of the model are the aircraft selections (binary) and the freight flows (continuous):

x_{sk} = the number of aircrafts of type K selected to serve a spider leg S.

y_{ij} = the amount of freight carried from city i to city j (1000 kgs).

In most cases the x_{sk} variables will be either zero or one (in case, any additional aircraft of a given type are rendering services along the spider leg s, then they will be allowed by declaring them as additional types, since the computer codes are composed of zero-one variables). The amount y_{ij} is necessarily sent on the unique path from i to j.

The capacitated model to the problem (1) requires additional denotations.

$N_{(i)}$= {(u,v) |the route from city u to city v includes the inbound arc from city i};

$T_{(i)}$= {(u,v) | the route from city u to v includes the out-bound arc to city i };

$leg_{(i)}$= the index of the spider leg that city i lies on.

$K_{(k)}$= the capacity of an aircraft of type k (1000 kgs).

The total capacity available is the same in the cases of inbound and outbound trips, since any aircraft chosen to serve $leg_{(i)}$ must make the trip in both directions.

Therefore, each city i adds two additional linear constraints to the model (1) given by;

$$\sum_{(u,v)N_{(i)}} Y_{uv} \leq \sum_{KM(leg_{(i)})} K_{(k)} x_{leg(i),k}$$

and

$$\sum_{(u,v)T_{(i)}} y_{uv} \leq \sum_{KM(leg_{(i)})} K_{(k)} x_{leg(i),k}$$

Such a problem can be solved by using branch-and-bound algorithm, which is discussed in the next chapter.

3. An integer program for decision support in the charitable disposition of excess inventory [13]:

Whenever inventory is sold for less than cost, donated to charity, or hauled to the landfill, net profit is negative for those transactions. Therefore the goal of disposing of inventories in any of these manners is to decrease the drain on the organization's overall net profit. This is accompanied by maximizing the income tax savings associated with the above alternatives.

Under certain circumstances, the charitable donation of these excess inventories can result in an income tax deduction allowed for dumping or lower than cost sales of inventory. The income tax savings from these donations can actually decrease the negative effects on net profit. The conditions under which charitable donation is the preferred alternative to selling or dumping are identified. Based on this information an integer program is developed to help managers determine the number of specific excess items to donate to charity.

The goal or objective of the decision to dispose of the excess inventory items should be to maximize the contribution to the net profit of the organization. Assuming the organization is opposed to dumping the items of excess inventory, the objective function should include the alternatives of clearance sale or charitable donation for each type of excess inventory item. It must also be assumed that the organization has taxable income prior to any charitable deductions. This assumption is made because only organizations liable for income tax can benefit from income tax savings.

For each inventory item types i that is sold at a clearance price, an ordinary income transaction occurs. The net profit that results from this ordinary income transaction is

$$[P_i - B_i - (P_i)(E_i)](1-t)$$

where,

P_i = the clearance price of item i.

B_i = the basis or cost of goods sold per unit of item i.

E_i = the selling expense of i, expressed as percent of P_i.

t = the effective tax rate of the organization.

Any items donated to charity do not produce ordinary income transactions. Their contribution to the net profit of the organization is the result of the income tax savings on the charitable deduction reduced by the basis of the item. The net profit of any item i that is denoted to charity is the lesser of

$$0.5(FMV_i - B_i)t - B_i(1-t) \text{ or } 2B_it - B_i$$

where FMV_i is the fair market value of item i.

If x_{1i} is the number of inventory items i sold at clearance price P_i and x_{2i} is the number of inventory items i donated to charity, the objective function of the integer program becomes:

$$\text{Max} \sum [\{P_i - B_i - (P_i)(E_i)\}(1-t)]x_{1i} + \min[\{0.5(FMV_i - B_i)t - B_i(1-t)\}x_{2i}, \{2B_itB_i\}x_{2i}]$$

The constraints on the decision will include the number of each type of excess inventory items and the annual limit on charitable deductions. Each type of excess inventory item i will have a constraint such as

$$x_{1i} + x_{2i} = I_i$$

where I_i = the amount of excess inventory item i.

1.5 Methods for solving integer programming problem:

The principal approaches for solving integer programming problems are (a) cutting plane techniques and (b) enumerative methods. We briefly describe these while the details of the algorithms are left to the next chapter.

(a). Cutting plane techniques:

The cutting plane technique consists of the derivation of new constraints (or cuts) so as to whittle the LP feasible region down to one whose optimal vertex is integer in the integer constrained variables.

Gomory, Ralph E. [24], first suggested such a method for solving an All-Integer programming problem and latter in 1960 he extended the procedure to cover a generalized case of mixed integer programming problem. Since then a number of advances have been made in cutting plane theory by Ghandforoush and Austin [21], Bell [10] , Sherali [39], Charnes and Granot [12], Jeroslow [28], and others.

(b). Enumerative methods-

The intent here is to enumerate, either explicitly or implicitly, all possible solution candidates to the mixed integer or integer program, the feasible solution which maximizes the objective function is optimal. Enumerative algorithm show that certain related integer points cannot yield improved solutions; that is via criteria which are contrived from the integrality and the constraint requirements, to implicitly enumerate large numbers of points.

Land and Doig [31] in 1960 proposed an enumerative technique for the general mixed integer program. Little, Murty, and others used the same idea in

developing an for the traveling salesman problem. In 1965 Balas[7] proposed an enumerative algorithm scheme for solving zero-one integer program.

1.6 NP- Hard problems:

Integer programs belong to a class of problems known as NP-hard. We may somewhat loosely think of NP as meaning "not polynomial". This means that there is no known algorithm of solving these problems such that the computational effort at worst increases as a polynomial in the problem size. For our purposes, we shall say that an algorithm runs in polynomial time if there is a positive constant k, such that the time to solve a problem of size n is proportional to n^k. For example, sorting a set of n numbers can easily be done in (polynomial) time proportional to n^2, where as solving an integer program in n zero/one variables may, in worst case, (exponential) time proportional to 2^n. There may be faster way but no one has published an algorithm for integer program that is guaranteed to take polynomial time on every problem presented to it. The terms NP-complete and P-complete apply to problems that can be stated as "yes/no" or feasibility problems. The yes/no variation of an optimization problem would be a problem of the form: Is there a feasible solution to this problem with cost less-than-or-equal-to 1250. In an optimization problem, we want a feasible solution with minimum cost. Khachian (1979) showed that the feasibility version of LP is solvable in polynomial time. So we say LP is in P. Integer programming stated in feasibility form, and a wide range of similar problems, belong to a class of problems called NP-complete. These problems have the feature that it is possible to convert any one of these problems into any other NP-complete problem in time that is polynomial in the problem size. Thus, if we can convert problem A into problem B in polynomial time, and then solve B in polynomial time, we then have a way of solving A in polynomial time.

CHAPTER II

METHODS OF INTEGER PROGRAMMING

2.1 INTRODUCTION

While dealing with an integer linear programming, we first solve the linear programming by removing the integer restrictions on **x**, and then the integer solutions are obtained by applying post optimal analysis with respect to integer restrictions.

Various methods have been developed for the solution of an integer linear programming (ILP) problem. These can broadly be divided into two classes, viz., **cutting plane techniques** and **branch-and-bound techniques**. In the present chapter we discuss the techniques developed in these classes.

2.2 CUTTING PLANE TECHNIQUES

The general intent of cutting plane algorithms is to deduce supplementary inequalities from the integrality and constraint requirements, which eventually produce a linear program, whose optimal solution is integer in the integer constrained variables.

The constraint generation idea was proposed by Dantzig, Fulkerson and Johnson [16] in 1954 in their work on the travelling salesman problem, and then in 1957 by Markowitz and Manne [34]. However, in 1958 Gomory [22] developed the first cutting plane algorithm applicable to any integer program. Shortly afterward, Gomory [41] and Beale [9] generalized Gomory's results to the mixed integer case. In 1960 Gomory [30] produced a second cutting plane algorithm for the integer program which requires only additions and subtractions in computation (an "all-integer" technique). All of the above methods maintain linear programs.

which are dual feasible, and are therefore often classified as dual cutting plane algorithms.

Glover [26] and Young [43] developed cutting plane algorithms for the integer program, which maintain linear problems that are primal feasible. Since primal feasible integer solutions are successively produced, the technique is referred to as a primal cutting plane algorithm.

.2.2.1 Dual-fractional cut:

This method concerns itself with a cutting plane algorithm for the integer program, which utilizes the dual simplex method and allows fractional numbers in computation. The basic approach for this integer program is as follows:

Step 1- Starting with an all integer tableau; solve the integer program as a linear one. If it is infeasible, so is the integer problem – terminate. If the optimal solution is all integers, the integer program is solved – terminate.

Step 2- Derive a new inequality constraint (or 'cut') from the integrality and constraint requirement, which cut off the (current) optimal point (i.e. makes the linear programming solution infeasible) but does not eliminate any integer solution. Add the new inequality to the bottom of the simplex tableau, which then exhibits primal infeasibility.

Step 3- Reoptimize using the dual (lexicographic) simplex method. If the new linear program is infeasible, the integer program has no solution – terminate. If the new optimum is in integers, the integer program is solved – terminate. Otherwise go to step 2.

2.2.2 Dual all-integer cut:

The dual all-integer method is a direct extension of the classical dual simplex algorithm. The essential difference is that the pivot row in the all integer algorithm

is generated at each iteration and ensures a -1 pivot. Since the technique employs the dual simplex method and maintains all integer tableaux, it is referred to as "dual all-integer".

The initial tableau is assumed to be all integer and lexicographic dual feasible. Hence successive tableaux are also all integer and lexicographic dual feasible. The primal integer solution proceeds towards feasibility, and since dual feasibility is maintained, optimality is reached when it is attained. The steps for this method are as follows:

Step 1- Start with an all integer simplex tableau, which contains a lexicographic dual feasible solution.

Step 2- Select a primal infeasible row v (i.e. $a_{v0} < 0, v \neq 0$). If none exist, the tableau exhibits the optimal integer solution – terminate. Go to step 3.

Step 3- Designate the pivot column α_p (p = 1,2,...,n) to be the lexicographically smallest among those having $a_{vj} < 0$. If none exist (i.e. $a_{vj} \geq 0$ for j = 1,2,...,n) there is no integer feasible solution – terminate. Go to step 4.

Step 4- Derive an all integer inequality from row v, which is not satisfied at the current primal solution. It must also have a -1 coefficient in column α_p. Adjoin it to the bottom of the tableau and label it the pivot row. Perform a dual simplex pivot operation and return to step 2.

The form of the cut:

Let the generating row be any row v of the

$$x_v = a_{v0} + \sum_{j=1}^{n} a_{vj}(-x_{J(j)})$$

Then the all integer cut is:

$$x' = \left[\frac{a_{v0}}{\lambda}\right] + \sum_{j=1}^{n}\left[\frac{a_{vj}}{\lambda}\right](-x_{J(j)}) \geq 0$$

where x' is a nonnegative Gomory slack variable and [y]means the largest integer $\leq y$. λ is a positive number found by rules below.

The rules for finding λ :

Step 1- With v as the generating row, let α_p be the lexicographically smallest column among those having $a_{vj} < 0$ (j = 1,2,...n).

Step 2- Let u_p=1, and for every $j \geq 1 (j \neq p)$ with $a_{vj} < 0$, let u_j be the largest integer such that $\left(\frac{1}{u_j}\right)\alpha_j \succ \alpha_p$.

Step 3- For each $a_{vj} < 0$ $(j \geq 1)$, set $\lambda_j = \frac{-a_{vj}}{u_j}$.

Note that λ_j is not necessarily an integer.

Step 4- Set λ = maximum λ_j. Note that $\lambda \geq \lambda_p = \frac{-a_{vp}}{u_p} \geq 1$, since u_p=1 and $-a_{vp}$ is a positive integer.

2.2.3 Primal all-integer cut:

The primal all-integer integer programming algorithm is an extension of the primal simplex method. Specifically, the procedure requires an all integer primal feasible initial tableau. It adds a Gomory cut at each iteration, starting with the very first so as to maintain an all integer tableau and primal feasibility. When dual

feasibility is reached the tableau is integer optimal and the integer program is solved.

Suppose the tableau is primal feasible and all integer. That is, each a_{ij} is integer and $a_{n+1} \geq 0 \quad (j = 1,2,\ldots m)$. If the dual problem is not feasible, there are column indexed by j $(j \geq 1)$ with $a_{0j} < 0$. Suppose, we let the pivot column correspond to the most negative of these, say a_{0p}. To determine a pivot row, we find a row satisfying $\left[\frac{a_{i0}}{a_{ip}}\right] \leq \theta_p \ , \ a_{ip} > 0$

Let the equation of such a row be

$$x = a_{i0} + \sum_{j=1}^{n}\left[\frac{a_{ij}}{\lambda}\right](-x_j) \geq 0.$$

Let the above equation serve as a generating row from which the Gomory all integer cut is derived as

$$s = \left[\frac{a_{i0}}{\lambda}\right] + \sum_{j=1}^{n}\left[\frac{a_{ij}}{\lambda}\right](-x_j) \geq 0.$$

The algorithm for the method is as follows:

Step 1- Start with a primal feasible all integer tableau. If such a tableau cannot be found the integer program is infeasible – terminate. Go to step 2.

Step 2- Find the pivot column indexed by p using $a_{0p} = \min_{a_{0j}<0} imum \ \ a_{0j} \quad (j \geq 1)$

If $a_{0j} \geq 0 \quad (j = 1,2,\ldots n)$, the tableau is integer optimal – terminate. Go to step 3.

Step 3- Find the row indexed by v utilizing $\theta_p = \frac{a_{v0}}{a_{vp}} = \min_{a_{ip}} imum \ \left\{\frac{a_{i0}}{a_{ip}}\right\}$.

Arbitrarily break ties. If $a_{ip} \leq 0 \quad (i = 1,2,\ldots,n+m)$, the integer program has an unbounded solution – terminate. Go to step 4.

Step 4- From a row i with $a_{ip} > 0$ that satisfies $\left[\frac{a_{i0}}{a_{ip}}\right] \leq \theta_p$, generate a Gomory all integer cut. Set the parameter λ in the cut equal to a_{ip}; let the derived row be the pivot row. Perform a primal simplex pivot step and return to step 2.

2.3 NAZ cut for integer programming [6]:

Here a new type of cut (termed as NAZ cut) has been proposed that reduces the feasible region of an integer program considerably. The procedure and an example are given to illustrate the cut.

Procedure for solving the problem using NAZ cut:

Step 1- Solve the given problem as LPP using simplex or dual simplex method.

Step 2- If this solution is integer, stop. Otherwise, round off the non-integer solution to the nearest integers.

Step 3- Find the minimum perpendicular distance from the integer point, which is inside the feasible region on the objective function curve passing through the non-integer solution. Derive NAZ cut passing through this point and parallel to the objective function curve.

Step 4- Use branch-and-bound or cutting plane method to find the integer optimum.

Example:

Maximize $z = 2x_1 + 3x_2$

Subject to $5x_1 + 2x_2 \leq 15$

$3x_1 + 5x_2 \leq 15$

$x_1, x_2 \geq 0$ and integer.

After solving this problem as a non-integer problem by using simplex method we get the non-integer solution as:

$x_1 = 2.37, x_2 = 1.58,$ and $z = 9.48$

So we round off the non-integer solution to the nearest four integer points as (2,2), (3,2), (3,1), and (2,1). Now calculate the perpendicular distances from these points by using the distance formula.

The distance from the point (2,2) is $\frac{-0.57}{\sqrt{13}}$

The distance from the point (3,2) is $\frac{-2.57}{\sqrt{13}}$

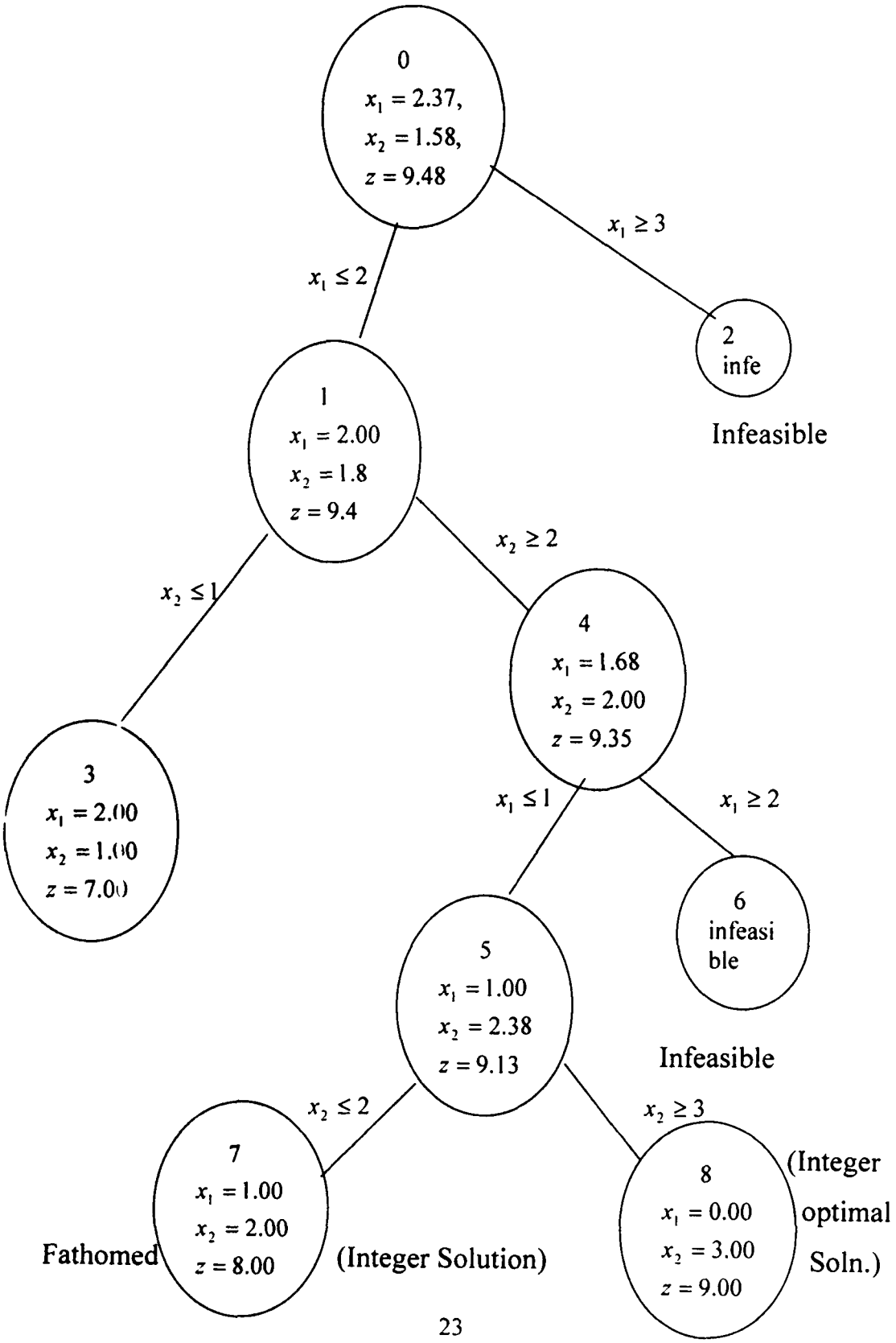
The distance from the point (3,1) is $\frac{+0.43}{\sqrt{13}}$

The distance from the point (2,1) is $\frac{+2.43}{\sqrt{13}}$

We discard those points for which distance is negative and check whether the constraints are satisfied for the points for which the distance is positive. If the constraints are not satisfied then discard that point. Now we are left with only one point (2,1), which is in the feasible region.

Now we derive NAZ cut passing through the integer point (2,1) as $2x_1 + 3x_2 \geq 7$ and solving the new problem by using branch-and-bound method we get the optimal integer solution to the problem as

$x_1 = 0, x_2 = 3,$ and $z = 9.$


0
$x_1 = 2.37$,
$x_2 = 1.58$,
$z = 9.48$
$x_1 \geq 3$
$x_1 \leq 2$
2
infe
Infeasible
1
$x_1 = 2.00$
$x_2 = 1.8$
$z = 9.4$
$x_2 \geq 2$
$x_2 \leq 1$
4
$x_1 = 1.68$
$x_2 = 2.00$
$z = 9.35$
3
$x_1 = 2.00$
$x_2 = 1.00$
$z = 7.00$
$x_1 \leq 1$
$x_1 \geq 2$
6
infeasi
ble
5
$x_1 = 1.00$
$x_2 = 2.38$
$z = 9.13$
Infeasible
$x_2 \leq 2$
$x_2 \geq 3$
7
$x_1 = 1.00$
$x_2 = 2.00$
$z = 8.00$
Fathomed
(Integer Solution)
8
$x_1 = 0.00$
$x_2 = 3.00$
$z = 9.00$
(Integer
optimal
Soln.)

2.4 THE BRANCH-AND-BOUND TECHNIQUE:

Because any bounded pure IP problem has only a finite number of feasible solutions, it is natural to consider using some kind of enumeration procedure for finding an optimal solution. But unfortunately, this finite number can be, and usually is, very large. Therefore, it is imperative that any enumeration procedure be cleverly structured so that only a tiny fraction of the feasible solution actually needs to be examined. Such approach is provided by the branch-and-bound technique. This technique and variations of it have been applied with some success to a variety of operations research problems, but it is, especially well known for its application to IP problems.

The basic concept underlying the branch-and-bound technique is to divide and conquer. Since the original "large" problem is too difficult to be solved directly it is divided into smaller and smaller sub problems until these sub problems can be conquered. The dividing (branching) is done by partitioning the entire set of feasible solutions into smaller and smaller subsets. These conquering (fathoming) is done partially by bounding how good the best solution in the subset can be, and then discarding the subset if its bound indicates that it can not possibly contains an optimal solution for the original problem.

This method can be applied to mixed, as well as pure integer programming problems. For definiteness, suppose the model is stated as

$$\text{Maximize } \sum_{j=1}^{n} c_j x_j \qquad \ldots\ldots(1)$$

$$\text{Subject to } \sum_{j=1}^{n} a_{ij} x_j \le b_i \quad \text{for i} = 1,2,\ldots,m \qquad \ldots\ldots(2)$$

$$x_j \text{ integer-valued for j} = 1,2,\ldots,p(\le n) \qquad \ldots\ldots(3)$$

$$x_j \geq 0 \text{ for } j = p+1,\ldots,n. \qquad \ldots\ldots(4)$$

In addition, assume that, for each integer-valued variable, we can provide lower and upper bounds that surely include the optimal values-

$$L_j \leq x_j \leq U_j \quad \text{for } j = 1,2,\ldots,p. \qquad \ldots\ldots(5)$$

Usually $L_j = 0$, but it need not.

The idea of branch-and-bound algorithm stems from the following elementary operat on.

Consider any variable x_j, and let i be some integer value, where $L_j \leq I \leq U_{j-1}$. Then an optimal solution to (1) through (5) will also satisfy either the linear constraint

$$x_j \geq I+1 \qquad \ldots\ldots(6)$$

or the linear constraint $x_j \leq I$. To illustrate how this dichotomy can be used, suppose we ignore the integer restriction (3) and find that an optimal linear programming solution to (1), (2), (4) and (5) indicates that $x_1 = 1\frac{2}{3}$. Then formulate and solve two more linear programs. Each of these still contains (1), (2) and (4). But (5) for j =1 is modified in one problem to be $2 < x_1 < U_1$, and in the other to be $L_1 \leq x_1 \leq 1$. Suppose further that each of these two problems has an optimal solution that satisfies the integer restriction (3). Then the solution that has the larger value for the objective function is indeed optimal for the original integer programming problem. Usually, on (or both) of these problems has no optimal solution that satisfies (3); hence, additional computations may be required. The algorithm below specifies how to apply the dichotomy (6) and (7) in a systematic manner to eventually obtain an optimal solution.

The method: At any iteration t, we have a lower bound, say, x_0^t for the optimal value of the objective function. To keep the exposition simple, assume that at the first iteration, x_0^1 either is strictly less than the optimal value, or equals the value of the objective function for a feasible solution that we have recorded. If worse comes to worst, we can let, $x_0^1 = -\infty$, if we have no information at all about the problem. In addition to a lower bound, we have a master list of linear programming problems that must be solved. The only differences among these comprise revisions in the bounds (5). At iteration 1, the master list contains a single problem consisting of (1), (2), (4) and (5).

The procedure of iteration t is:

Step 1- Terminate the computations if the master list is empty. Otherwise, remove a linear programming problem from the master list.

Step 2- Solve the chosen problem. If it has no feasible solution, or if the resultant optimal value of the objective function x_0 is less than or equal to x_0^t, then let $x_0^{t+1} = x_0^t$, and return to step 1. Otherwise, proceed to step 3.

Step 3- If the obtained optimal solution to the linear programming problem satisfies the integer constraints, then record it, let x_0^{t+1} be the associated optimal value of the objective function x_0, and return to step 1. Otherwise proceed to step4.

Step 4- Select any variable x_j, for j = 1,2,..., p, that does not have an integer value in the obtained optimal solution to the chosen linear programming problem. Let b_j denote this value, and $[b_j]$ signify the largest integer less than or equal to b_j. Add two linear programming problems to the master list. These two problems are identical with the problem chosen in step 1, except that in one, the lower bound on x_j is replaced by $[b_j]+1$, and in the other, the upper bound on x_j is replaced by

$[b_j]$. Let $x_0^{t+1} = x_0^t$, and return to step 1.

At termination if we have recorded a feasible solution yielding x_0^t, it is optimal; otherwise, no feasible solution exists.

The process at step 1 is called branching because it involves the selection of a linear programming problem for further consideration. The process at step 2 is known as relaxation; here we solve linear programming problem ignoring (relaxing) the integer –value constraints. Since such a problem is less constrained than the same problem with the integer stipulations in force, the linear programming objective function value is at least as large as that for the corresponding integer problem. Thus if the linear programming solution does not yield an objective function value larger than the current lower bound, we need not consider the problem further. In this event, the problem is said to have been fathomed. The process at step 4 is known as separation. Here a "parent" linear programming problem with an optimal objective function larger than the current lower bound gives rise to two descendants. If we augment the procedure in step 4 to record on the master list with each descendant the optimal value of the objective function for the parent linear programming problem, then the largest of these values for all problems currently on the master list is an upper bound on the optimal objective function value for the integer problem. Thus, if we terminate the algorithm before the master list is empty, we can assess the improvement potential from the linear programming problems that remains on the list as compared to the best feasible solution that we have obtained so far.

An example will clarify the details of the procedure. Consider

Maximize $3x_1 + 3x_2 + 13x_3$ .....(8)

$$\text{Subject to} \quad \left.\begin{aligned} -3x_1 + 6x_2 + 7x_3 \le 8 \\ 6x_1 - 3x_2 + 7x_3 \le 8 \end{aligned}\right\} \qquad \text{.....(9)}$$

where each x_j must be a non-negative integer. Suppose we specify the bounds on each variable as

$$0 \le x_j \le 5 \text{ for } j = 1,2,3 \qquad \text{.....(10)}$$

As usual, let x_0 denote the value of the objective function. Find the optimal solution by inspection.

At iteration 1, let the lower bound be $x_0^1 = 0$, since all $x_j = 0$ is feasible. The master list contains only the linear programming problem (8), (9), and (10), which is designated as problem 1. Remove it in step 1, and in step 2 find the optimal solution.

Problem 1:

$$x_0 = 16, \quad x_1 = x_2 = 2\frac{2}{3}, \quad x_3 = 0 \qquad \text{.....(11)}$$

Since the solution is not integer valued, proceed from step 3 to step 4, and select x_1. Then since $[b_1] = \left[2\frac{2}{3}\right] = 2$, place on the master list

Problem 2: constraints (9) and

$$3 \le x_1 \le 5 \qquad 0 \le x_2 \le 5 \qquad 0 \le x_3 \le 5$$

Problem 3: constraints (9) and

$$0 \le x_1 \le 2 \qquad 0 \le x_2 \le 5 \qquad 0 \le x_3 \le 5$$

Returning to step 1 with $x_0^2 = x_0^1 = 0$, remove problem 2. Step 2 establishes that problem 2 has no feasible solution. Hence, put $x_0^3 = x_0^2 = 0$, and return to step 1.

Now remove problem 3 and obtain in step 2 the optimal solution;

$$x_0 = 15\frac{5}{7}, \quad x_1 = x_2 = 2. \qquad x_3 = \frac{2}{7} \qquad \text{.....(13)}$$

which is not integer-valued. Therefore, go from step 3 to step 4, where x_3 is selected.

Since $[b_3] = \left[\frac{2}{7}\right] = 0$, place on the master list:

Problem 4: constraints (9) and

$$0 \le x_1 \le 2 \qquad 0 \le x_2 \le 5 \qquad 1 \le x_3 5$$

Problem 5: constraints (9) and

$$0 \le x_1 \le 2 \qquad 0 \le x_2 \le 5 \qquad 0 \le x_3 \le 0$$

Observe that problem 4 and 5 differ from problem 3 only in the bounds on x_3. Returning to step 1 with $x_0^4 = 0$, remove problem 4. The optimal solution is

$$x_0 = 15, \qquad x_1 = x_2 = \frac{1}{3}, \qquad x_3 = 1.$$

This leads to step 4; suppose we select x_2, yielding, as a consequence,

Problem 6: constraints (9) and

$$0 \le x_1 \le 2 \qquad 1 \le x_2 \le 5 \qquad 1 \le x_3 \le 5$$

Problem 7: constraints (9) and

$$0 \le x_1 \le 2 \qquad 0 \le x_2 \le 0 \qquad 1 \le x_3 \le 5$$

Returning to step 1 with $x_0^5 = 0$, remove problem 6 so that problem 5 and 7 remain on the master list. We will discover in step that problem 6 has no feasible solution,

so return to step 1 with $x_0^0 = 0$. Now remove problem 7, giving the optimal solution;

$$x_0 = 14\frac{6}{7}, \qquad x_1 = x_2 = 0, \qquad x_3 = 1\frac{1}{7}$$

Because x_3 is fractional and $\left[1\frac{1}{7}\right] = 1$, this creates, in step 4,

Problem 8: constraints (9) and

$$0 \le x_1 \le 2 \qquad 0 \le x_2 \le 0 \qquad 2 \le x_3 \le 5$$

Problem 9: constraints (9) and

$$0 \le x_1 \le 2 \qquad 0 \le x_2 \le 0 \qquad 1 \le x_3 \le 1$$

Removing problem 8 at iteration 7 gives an indication of no feasible solution in step 2. Remove problem 9 at iteration 8 and observe that only the value of x_1 can still vary. So for step 2 find the optimal integer solution which is

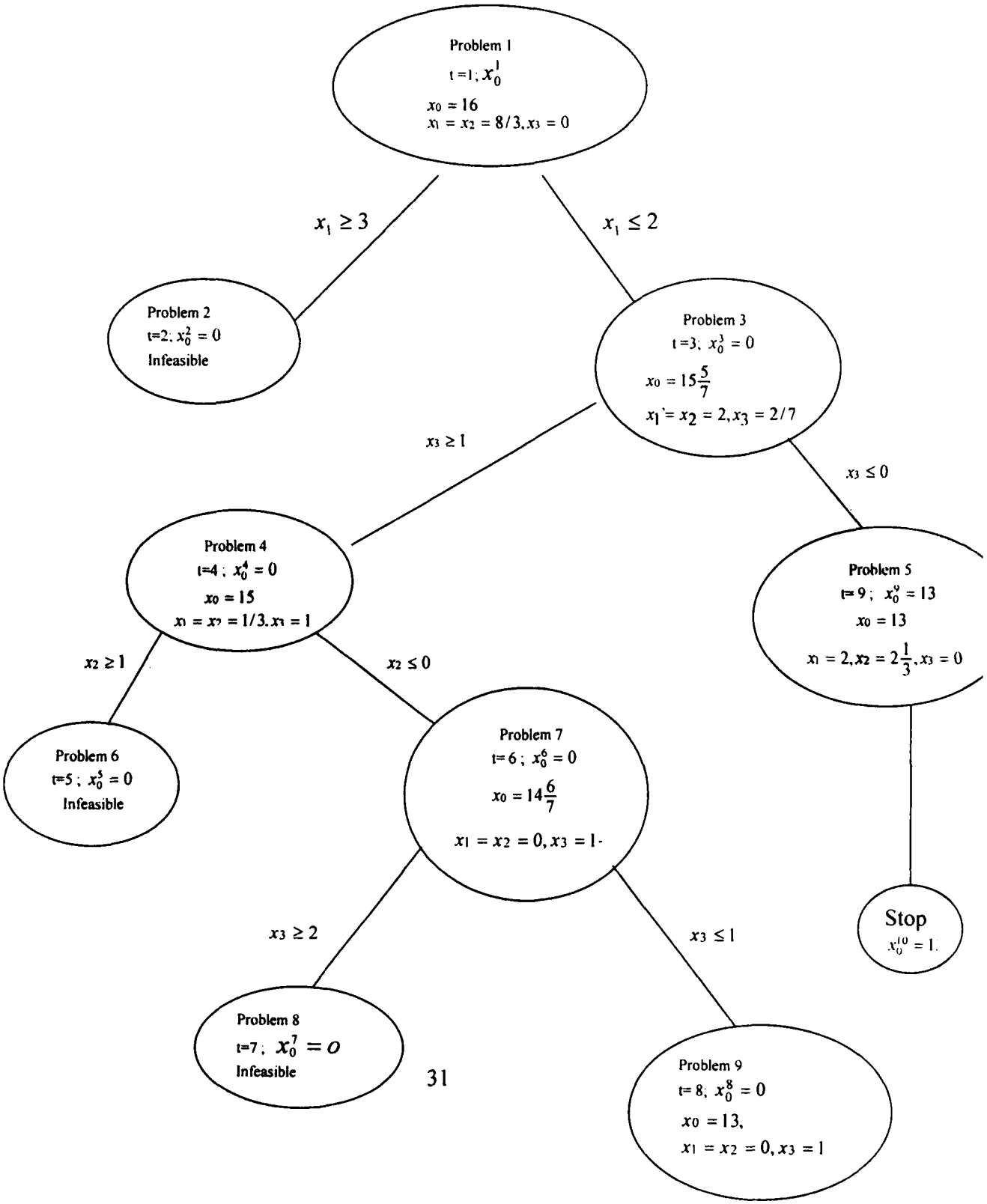
$$x_0 = 13, \qquad x_1 = x_2 = 0 \qquad x_3 = 1$$

Therefore at step 3, we record (19) and let $x_0^9 = 13$. Returning to step 1, we find that only problem 5 remains on the master list. The optimal linear programming solution is

$$x_0 = 13, \quad x_1 = 2, \quad x_2 = 2\frac{1}{3}, \quad x_3 = 0.$$

Since x_0 in (20) equals x_0^9, we return to step 1 and terminate the computations, as

the master list is now empty. The optimal solution to the integer programming problem is (19), which was recorded at iteration 8.



$$\text{Maximize } z = 9x_1 + 5x_2 + 6x_3 + 4x_4$$

$$\text{Subject to } \quad 6x_1 + 3x_2 + 5x_3 + 2x_4 \leq 10$$

$$x_3 + x_4 \leq 1$$

$$-x_1 + x_3 \leq 0$$

$$-x_2 + x_4 \leq 0$$

$$x_j \text{ is binary, } \quad j = 1, 2, 3, 4$$

We shall now describe in turn the three basic steps – branching, bounding, and fathoming and illustrate them by applying a branch-and-bound algorithm to the above example.

Branching:

When dealing with binary variables, the most straight forward way to partition the set of feasible solutions into subsets is to fix the value of one of the variables (say, x_1) at $x_1 = 0$ for one subset and $x_1 = 1$ for the other subset. Doing this for the above example divides the whole problem into the two smaller sub problems shown below.

Sub problem 1: $(x_1 = 0)$

$$\text{Maximize } z = 5x_2 + 6x_3 + 4x_4$$

$$\text{Subject to } \quad 3x_2 + 5x_3 + 2x_4 \leq 10$$

$$x_3 + x_4 \leq 1$$

$$x_3 \leq 0$$

$$-x_2 + x_4 \leq 0$$

x_j is binary, j = 1, 2, 3, 4

Sub problem 2: ($x_1 = 1$)

Maximize $z = 9 + 5x_2 + 6x_3 + 4x_4$

Subject to $3x_2 + 5x_3 + 2x_4 \leq 4$

$$x_3 + x_4 \leq 1$$

$$x_3 \leq 1$$

$$-x_2 + x_4 \leq 0$$

and x_j is binary, j = 2, 3, 4

Figure 1 portrays this dividing (branching) into sub problems by a tree with branches from the 'All' node (corresponding) to the two sub problems. This tree, which will continue "growing branches" iteration by iteration, is referred to as the solution tree for the algorithm. The variable used to do this branching at any iteration by assigning values to the variable is called the branching variable.

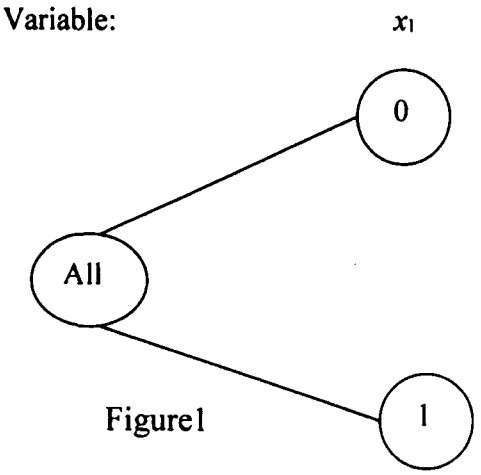


Figure1

One of these sub problems can be fathomed immediately, whereas the other sub problem will need to be divided further into smaller sub problems by setting $x_2 = 0$ or $x_2 = 1$, etc.

For each of these sub problems, we now need to obtain a bound on how good its best feasible solution can be. The standard way of doing this is to quickly solve a simpler relaxation of the sub problem.

To illustrate for the example, consider first the whole problem given by (1). Its LP-relaxation is obtained by deleting the last line of the model, but retaining the $x_j \le 1$ and $x_j \ge 0$ constraints. Using the simplex method to quickly solve this LP-relaxation yields its optimal solution, (x_1, x_2, x_3, x_4)= (5/6, 1, 0, 1), with z = $16\frac{1}{2}$. Therefore, $z \le 16\frac{1}{2}$ for all feasible solutions for the original BIP problem. This bound of $16\frac{1}{2}$ can be rounded down to 16, because all coefficients in the objective function are integer, so all integer solutions must have an integer value for z.

Bound for whole problem $z \le 16$. Now let us obtain the bounds for the two sub problems in the same way. Their LP-relaxation are obtained by replacing the constraints, x_j is binary for j = 2, 3, 4, by $0 \le x_j \le 1$ for j = 2, 3, 4. Applying the simplex method then yields their optimal solutions shown below

LP-relaxation of sub problem1: $(x_1, x_2, x_3, x_4) = (0,1,0,1)$ with z = 9.

LP-relaxation of sub problem 2: $(x_1, x_2, x_3, x_4) = (1,4/5,0,4/5)$ with z = $16\frac{1}{5}$.

The resulting bounds for the sub problems then are:

Bound for sub problem 1: $z \le 9$

Bound for sub problem 2: $z \leq 16$.

Figure 2 summarizes these results, where the numbers given just below the nodes are the bounds, and below each bound is the optimal solution obtained for the LP-relaxation.

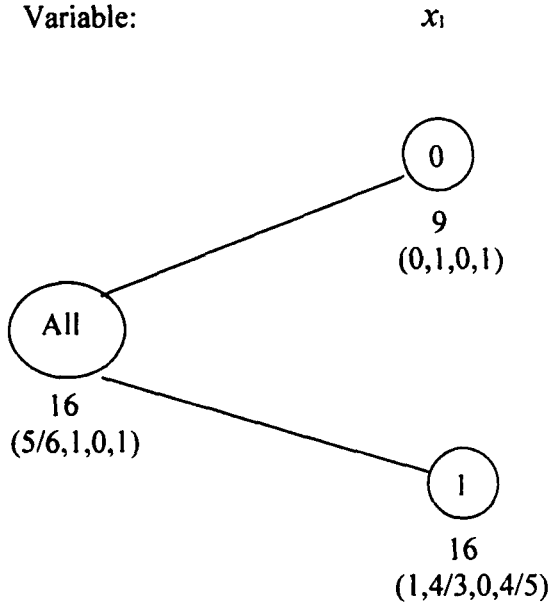


Figure 2

Fathoming:

A sub problem can be fathomed, and there by dismissed from further consideration in the three ways described below.

One way is illustrated by the results for sub problem 1 given by the $x_1 = 0$ node in figure 2. Note that the (unique) optimal solution for its LP- relaxation, (x_1, x_2, x_3, x_4)=(0,1,0,1), is an integer solution. Therefore, this solution must also be the optimal solution for sub problem 1 itself. This solution should be stored as the first incumbent (the best feasible solution found so far) for the whole problem, along with its value of z. This value is denoted by z*. So z* = 9 at this point.

Having stored this solution, there is no reason to consider sub problem 1 any further by branching from $x_1 = 0$ node, etc. Doing so could only lead to other feasible solutions that are inferior to the incumbent, and we have no interest in such solutions. Because it has been solved, we fathom sub problem 1 now.

The above results suggest a second key-fathoming test. Since $z^* = 9$, there is no reason to consider further any sub problem whose bound ≤ 9, since such a sub problem cannot have a feasible solution better than the incumbent. Stated more generally, a sub problem is fathomed whenever its bound $\leq z^*$.

This outcome does not occur in the current iteration of the example because sub problem 2 has a bound of 16 that is larger than 9.

The third way of fathoming is quite straightforward. If the simplex method finds that a sub problem's LP- relaxation has no feasible solutions, then the sub problem itself must have no feasible solutions, so it can be fathomed.

The summary of fathoming test is :

Test 1: Its bound $\leq z^*$.

or

Test 2: Its LP-relaxation has no feasible solution.

or

Test 3· The optimal solution for its LP-relaxation is integer. (If this solution is better than the incumbent, it becomes the new incumbent, and test 1 is reapplied to all unfathomed sub problems with the new larger z^*.)

Figure 3 summarizes the results of applying these three tests to sub problems 1 and 2 by showing the current solution tree. Only the sub problem 1 has been fathomed,

as indicated by the F(3) next to the $x_1 = 0$ node. The resulting incumbent also is identified below this node.

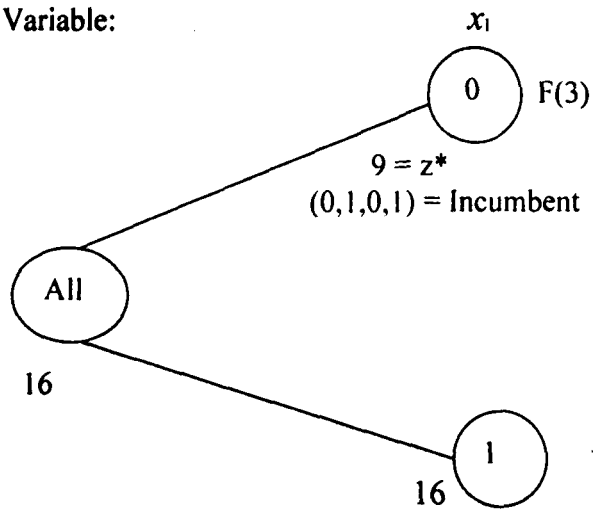


Figure 3

Completing the example-

The pattern for the remaining iterations will be quite similar to that for the first iteration described above except for the ways in which fathoming occurs.

Iteration 2: The only remaining sub problems correspond to the $x_1 = 1$ node in figure 3. So we shall branch from this node to create the two new sub problems given below.

Sub problem 3- $(x_1 = 1, x_2 = 0)$

Maximize $z = 9 + 6x_3 + 4x_4$

Subject to $5x_3 + 2x_4 \leq 4$

$x_3 + x_4 \leq 1$

$$x_3 \le 1$$

$$x_4 \le 0$$

and x_j is binary, j = 3, 4

Sub problem 4- $(x_1 = 1, x_2 = 1)$

Maximize $z = 14 + 6x_3 + 4x_4$

Subject to $5x_3 + 2x_4 \le 1$

$$x_3 + x_4 \le 1$$

$$x_3 \le 1$$

$$x_4 \le 0$$

and x_j is binary, for j = 3, 4

The LP-relaxation of these sub problems are obtained by replacing the constraints, x_j is binary for j = 3, 4 by $0 \le x_j \le 1$ for j = 3, 4. Their optimal solutions are:

LP-relaxation of sub problem 3: $(x_1, x_2, x_3, x_4) = (1, 0, 4/5, 0)$ with $z = 13\frac{4}{5}$.

LP-relaxation of sub problem 4: $(x_1, x_2, x_3, x_4) = (1, 1, 0, 1/2)$ with z = 16.

The resulting bounds for the sub problems are:

Bound for sub problem 3: $z \le 13$

Bound for sub problem 4: $z \le 16$

Note that both of these bounds are larger than $z^* = 9$, so fathoming test 1 fails in both cases. Test 2 also fails, since both LP-relaxations have feasible solutions.

Test 3 fails as well, because both optimal solutions include variables with non-integer values.

Figure 4 shows the resulting solution tree at this point.

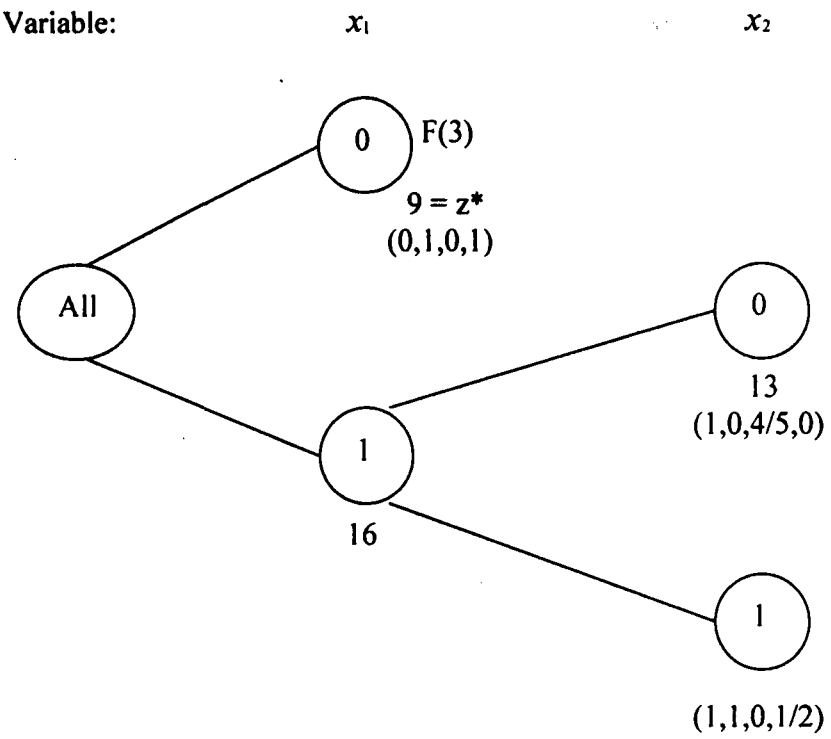


Figure 4

Iteration 3: Sub problem 4 $(x_1 = 1, x_2 = 1)$ has the larger bound (16>13), the next branching is done from the (x_1, x_2) = (1, 1) node in the solution tree, which creates the following new sub problems.

Sub problem 5: $(x_1 = 1, x_2 = 1, x_3 = 0)$

$$\text{Maximize } z = 14 + 4x_4$$

$$\text{Subject to } 2x_4 \leq 1$$

$$x_4 \leq 1$$

and x_4 is binary.

Sub problem 6: $(x_1 = 1, x_2 = 1, x_3 = 1)$

$$\text{Maximize } z = 20 + 4x_4$$

$$\text{Subject to } 2x_4 \leq -4$$

$$x_4 \leq 0$$

$$x_4 \leq 1$$

and x_4 is binary.

Forming their LP-relaxation by replacing x_4 is binary by $0 \leq x_4 \leq 1$, the following results are obtained.

LP-relaxation of sub problem 5: $(x_1, x_2, x_3, x_4) = (1, 1, 0, 1/2)$ with $z = 16$

LP-relaxation of sub problem 6: No feasible solution.

Bound for sub problem 5: $z \leq 16$

We now have a solution tree shown in figure 5.

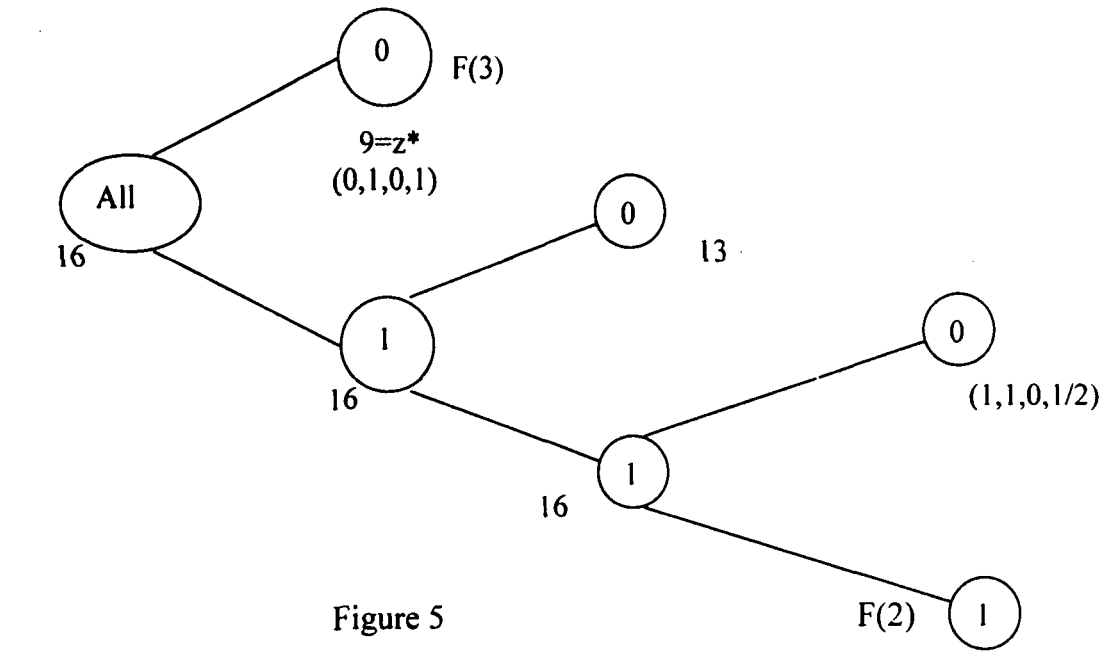


Figure 5

Iteration 4: The sub problems corresponding to nodes (1,0) and (1,1,0) in figure 5 remain under consideration, but the latter node was created more recently, so it is selected for branching from next. Since the resulting branching variable x_4 is the last variable, fixing its value at either 0 or 1 actually creates a single solution rather than a sub problems requiring fuller investigation. These single solutions are

$x_4 = 0$ $(x_1, x_2, x_3, x_4) = (1, 1, 0, 0)$ is feasible with Z=14,

$x_4 = 1$ $(x_1, x_2, x_3, x_4) = (1, 1, 0, 1)$ is feasible.

Formally applying the following tests, the first solution passes test 3 and the second solution passes test 2. Furthermore this possible first solution is better than the incumbent (14>9) so it becomes the new incumbent, with z*=14. Because a new incumbent has been found, we now reapply following test 1 with the new

larger value of z* to the only remaining sub problem, the one at node (1, 0)

Sub problem 3: Bound = 13 ≤ z* = 14

Therefore, this sub problem now is fathomed; we now have the solution tree shown in fig. 6. Note that there are no remaining (unfathomed) problems. Consequently, the optimality test indicates that the current incumbent

$$(x_1, x_2, x_3, x_4) = (1, 1, 0, 0)$$

is optimal, so we are done.

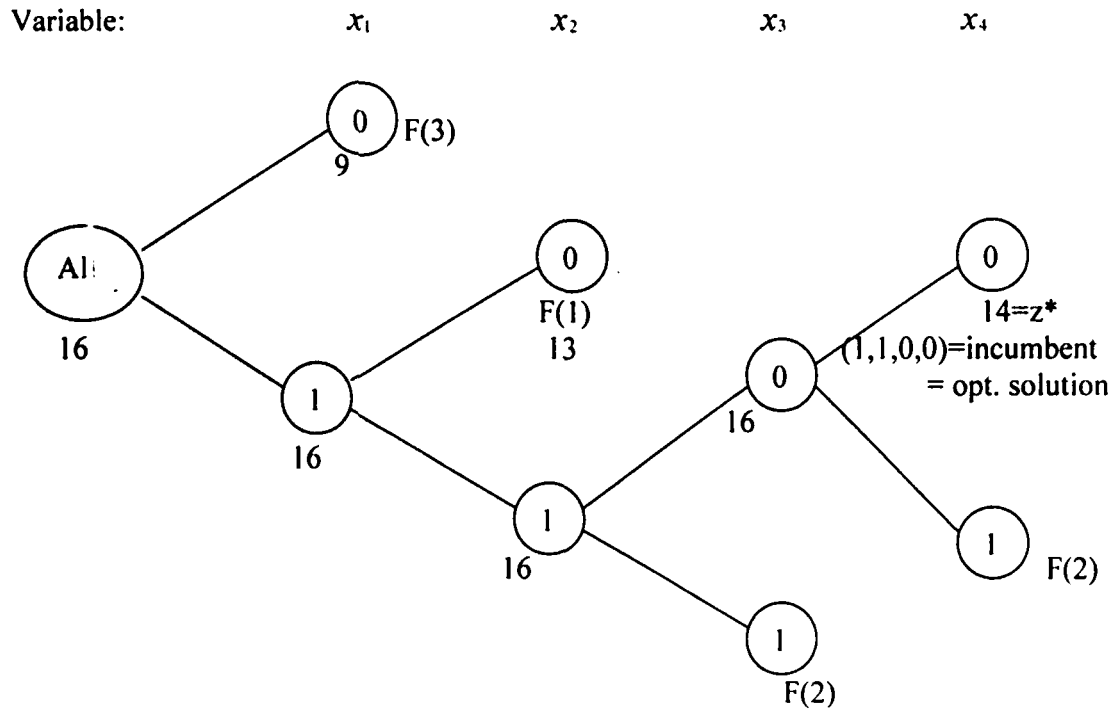


Figure 6

CHAPTER III

MIXED INTEGER PROGRAMMING PROBLEM

3.1 INTRODUCTION

In some integer programming problems, one or more of the decision variables must be integer while the other decision variables are allowed to take any integer or non-integer value. When on integer programming problem allows fractional vales to one subset of variables while requiring integer values to another subset of variables in the optimal solution, it is referred to as a Mixed integer programming problem.

3.2 MIXED INTEGER CUT

The solution method of mixed integer programming problems is similar in most aspects to that of regular integer programming problems. As in the case of regular integer programming, an optimal solution without integer restrictions is initially obtained by the simplex method. The Gomory constraint is formulated by taking the integer restricted basic variable, which has the largest fractional value in the optimal solution to the ordinary linear programming problem.

Suppose that the basic variable x_i has a fractional value in the optimal simplex tableau of the ordinary linear programming problem, although it is restricted to take on integer value. If the non-basic variables are denoted by x_j the basic variable has the following relationship with the non-basic variable:

$$x_i = \hat{b}_i + \sum_{j=1}^{n} \hat{a}_{ij} x_j \qquad (\text{for } i = 1,2.....,m)$$

or $$x_i + \sum_{j=1}^{n} (-\hat{a}_{ij} x_j) = \hat{b}_i$$

where $\hat{a}_{ij}$ = transformed values of the original coefficients a_{ij}.

$\hat{b}_i$ = transformed values of the original constants b_i.

The coefficients $\hat{a}_{ij}$ and the constants $\hat{b}_i$ are divided into two parts, a non-negative fraction and an integer :

$$\hat{a}_{ij} = f(\hat{a}_{ij}) + \text{integer}$$

$$\hat{b}_i = f(\hat{b}_i) + \text{integer.}$$

The new Gomory constraint for mixed integer programming problem is expressed as

$$\sum_{j=1}^{n} f'(\hat{a}_{ij})x_j = f(\hat{b}_i) = \text{integer (for } i = 1,2,....,m).$$

Therefore, $\sum_{j=1}^{n} f'(\hat{a}_{ij})x_j \geq f(\hat{b}_i)$

Where $f'(\hat{a}_{ij})$ = newly obtained non-negative fractional values.

By producing a Gomory slack variable $\hat{s}$ and rearranging the Gomory constraint inequality, the Gomory –constraint equation obtained is

$$\hat{s} = -f(\hat{b}_i) + \sum_{j=1}^{n} f'(\hat{a}_{ij})x_j \text{ (for } i = 1,2,....,m)$$

The newly introduced non-negative fractional values $f'(\hat{a}_{ij})$ are obtained by using the following specific rules. The value of $f'(\hat{a}_{ij})$ is determined by whether or not the non-basic variables in the Gomory constraint are required to take on integer values. The rules are

1. If the zero variable x_j is restricted to take on an integer value, then

$$f'(\hat{a}_{ij}) = f(a_{ij}) \text{ (for } f(\hat{a}_{ij}) \leq f(\hat{b}_i)$$

$$f'(\hat{a}_{ij}) = \frac{f(\hat{b}_i)}{1 - f(\hat{b}_i)}\left[1 - f(\hat{a}_{ij})\right] \text{ for } f(\hat{a}_{ij}) > f(\hat{b}_i)$$

2. If the zero variable x_j is not restricted to take on an integer value, then

$$f'(\hat{a}_{ij}) = \hat{a}_{ij} \quad \text{for } \hat{a}_{ij} \geq 0$$

$$f'(\hat{a}_{ij}) = \frac{f(\hat{b}_i)}{1 - f(\hat{b}_i)}(-\hat{a}_{ij}) \quad \text{for } \hat{a}_{ij} < 0.$$

Once the Gomory constraint equation is determined, it is augmented to the optimal simplex tableau of the ordinary linear programming problem. The computational procedures for the remaining part of the problem are exactly the same as those for the regular integer-programming problem.

3.2.1 Theorem: – The Gomory cut does not remove any mixed integer solution from the feasible region.

Proof: Let the source row be

$$x_q = a_{q0} + \sum_{j=1}^{n} a_{qj}(-x_{J(j)}) \qquad \ldots\ldots (1)$$

Where $f_{q0} = a_{q0} - \lfloor a_{q0} \rfloor > 0$ ..... (2)

Since x_q should be an integer variable

We have $x_q \equiv 0 \pmod 1$.

Hence $0 \equiv a_{q0} + \sum_{j=1}^{n} a_{qj}(-x_{J(j)}) \pmod 1$ ..... (3)

Now $a_{q0} > 0$ and we may add $0 \equiv -\lfloor a_{q0} \rfloor \pmod 1$ to (3) to reduce a_{q0} to its fractional part f_{q0}.

Thus, (3) may be written as

$$0 \equiv f_{q0} + \sum_{j=1}^{n} a_{qj}(-x_{J(j)}) \pmod 1 \qquad \ldots\ldots (4)$$

(4) may be written as

$$\sum_{j=1}^{n} a_{qj} x_{J(j)} = f_{q0} \pmod 1 \qquad \ldots\ldots (5)$$

Now suppose the L.H.S of (5) is positive. Thus for it to differ from f_{q0} by an integer amount, it must be equal f_{q0} or, $1+f_{q0}$ or $2+f_{q0}$ etc. Hence, we have,

$$\sum_{j=1}^{n} a_{qj} x_{J(j)} \geq f_{q0}$$

Then for any feasible solution $x_{J(j)} \geq 0$. $j = 1,2,\ldots\ldots,m$.

$$\sum_{j\in P} a_{qj} x_{J(j)} \geq \sum_{j=1}^{n} a_{qj} x_{J(j)} \geq f_{q0} \qquad \ldots\ldots (6)$$

Where $P = \{j/the\ coeff.\ of\ x_{J(j)} > 0\ ,\ j = 1,2,\ldots\ldots,n\}$

Now let the L.H.S. of (5) be negative. Then it has to equal $-1+f_{q0}$, $-2+f_{q0}$ etc. or in this case,

$$\sum_{j=1}^{n} a_{qj} x_{J(j)} \leq -1 + f_{q0}\ .$$

Then for any feasible solution $x_{J(j)} \geq 0, j = 1,2,\ldots\ldots,n$.

$$\sum_{j\in N} a_{qj} x_{J(j)} \leq \sum_{j=1}^{n} a_{qj} x_{J(j)} \leq -1 + f_{q0} \qquad \ldots\ldots (7)$$

Where $N = \{j/the\ coeff.of\ x_{J(j)} \leq 0\ ,\ j = 1,\ldots\ldots,n\}$

Multiplying the outside terms of (7) by negative number $f_{q0}/(-1+f_{q0})$ gives

$$\sum_{j\in N}\left(\frac{f_{q0}}{-1+f_{q0}}\right) a_{qj} x_{J(j)} \geq f_{q0} \qquad \ldots\ldots (8)$$

Now, the L.H.S. of (5) is either positive or negative .It cannot be zero since f_{q0} is a proper fraction. Hence at least one of the inequalities (6) and (.7) is true. But in any feasible solution each of these has a non-negative L.H.S.

Therefore, we have

$$\sum_{j\in P} a_{qj} x_{J(j)} + \sum_{j\in N}\left(\frac{f_{q0}}{-1+f_{q0}}\right) a_{qj} x_{J(j)} \geq f_{q0} \qquad \ldots\ldots (9)$$

After adding a non-negative Gomory slack variable x_{n+m+k}, (9) is

$$x_{n+m+k} = -f_{q0} + \sum_{j\in P} a_{qj} x_{J(j)} + \sum_{j\in N}\left(\frac{f_{q0}}{-1+f_{q0}}\right) a_{qj} x_{J(j)} \geq 0 \qquad \ldots\ldots (10)$$

From the derivation it is clear that (10) does not eliminate any mixed integer solution from the feasible region.

3.2.2 NUMERICAL ILLUSTRATION:

Maximize $-4x_1 - 5x_2 = x_0$

Subject to $-x_1 - 4x_2 \le -5$

$-3x_1 - 2x_2 \le -7$

$x_1, x_2 \ge 0$

& x_0, x_1 are integer.

The optimal simplex tableau for the above problem is

		$-x_4$	$-x_3$
Z	-112/10	11/10	7/10
x_1	18/10	-4/10	2/10
x_2	8/10	1/10	-3/10
x_3	0		-1
x_4	0	-1	0
x_5	-8/10	-16/10	-2/10

$$f_{10} = \frac{18}{10} - \left[\frac{18}{10}\right] = \frac{8}{10}$$

Since $a_{oj} = a_{11} \le 0$ & $x_{J(j)}$ is continuous.

$$\therefore g_{11} = \frac{8/10}{8/10 - 1}\left(\frac{-4}{10}\right) = \frac{16}{10}.$$

$$a_{12} > 0, \ \therefore g_{12} = \frac{2}{11}.$$

		$-x_5$	$-x_3$
Z	-188/16	11/16	9/16
x_1	2	-4/16	4/16
x_2	12/16	1/16	-5/16
x_3	0		-1
x_4	8/16	-10/16	2/16
x_5	0	-1	0
x_6	-4/16	-11/16	-9/16

$$f_{00} = \frac{-188}{16} - \left[\frac{-188}{16}\right] = \frac{4}{16}.$$

Since $a_{01} > 0, \therefore g_{01} = \frac{11}{16}$.

Also, $a_{02} > 0, \therefore g_{02} = \frac{9}{16}$.

		$-x_6$	$-x_3$
Z	-12	1	0
x_1	23/11	-4/11	5/11
x_2	8/11	1/11	-4/11
x_3	0		-1
x_4	8/11	-10/11	7/11
x_5	4/11	-16/11	9/11
x_6	0	-1	0
x_7	-1/11	-4/11	-5/11

$$f_{10} = \frac{23}{11} - \left[\frac{23}{11}\right] = \frac{1}{11}$$

$$g_{11} = \frac{1/11}{\left(1/11 - 1\right)}\left(\frac{-4}{11}\right) = \frac{4}{110}$$

$$g_{12} = 5/11.$$

		$-x_6$	$-x_7$
Z	-12	1	0
x_1	2	-2/5	1
x_2	4/5	6/50	-4/5
x_3	1/5	4/50	-11/5
x_4	3/5	-48/50	7/5
x_5	1/5	-76/50	9/5
x_6	0	-1	0
x_7	0	0	-1

The above tableau gives us the required optimal solution.

3.3 A BRANCH AND BOUND ALGORITHM FOR MIXED INTEGER PROGRAMMING

We shall consider the general MIP problem, where some of the variables (say I, of them) are restricted to integer values but the rest are ordinary continuous variables. For notational convenience, we shall order the variables so that the first I variables are the integer restricted variables. Therefore, the general form of the problem being considered is

Maximize $z = \sum_{j=1}^{n} c_j x_j$

Subject to $\sum_{j=1}^{n} a_{ij} x_j \le b_i$, for $i = 1,2,\ldots\ldots,m.$

and $x_j \ge 0$, for $j = 1,\ldots\ldots,n$.

x_j is integer, for $j = 1,\ldots\ldots\ldots\ldots,I\ (I \le n)$.

(when $I = n$, this problem becomes the pure IP problem.)

We shall describe a basic branch-and-bound for solving this problem, with a variety of refinements, has provided the standard approach to

MIP. The structure of this algorithm was first developed by R.J. Dakin.[15] , based on the pioneering branch-and-bound algorithm by A.H. Land and A.G. Doig. This algorithm is quite similar in structure to the Binary Integer programming (BIP) algorithm presented in the preceding chapter. In fact only four changes are needed in the BIP algorithm to deal with the generalization from binary to general integer variables and from pure IP to mixed IP.

One change involves the choice of the branching variable. Now, the only variables considered are the integer-restricted variables that have a non-integer value in the optimal solution for the linear programming relaxation of the current sub problem. Our rule for choosing among these variables is to select the first one in the natural ordering.

The second change involves the values assigned to the branching variables for creating the new smaller sub problems. Now, the general integer restricted variable could have a very large number of possible integer values, and it would be inefficient to create and analyze many sub problems by fixing the variable at its individual integer values. Therefore, what is done instead is to create just two new sub problems by specifying the two ranges of values for the variable. To spell out how this is done, let x_j be the current branching variable, and let x_j^* be its non-integer value in the optimal solution for the LP-relaxation of the current sub problem. Using a square bracket to denote

$\lfloor x_j^* \rfloor$ = greatest integer less than or equal to x_j^* .

The range of the values for the two new sub problems is

$x_j \le \lfloor x_j^* \rfloor$ and $x_j \ge \lfloor x_j^* \rfloor + 1$,

respectively. Each inequality becomes an additional constraint for that new sub problem.

The third change involves the bounding step. Before, with a pure IP problem and integer coefficients in the objective function, the value of the Z for the optimal solution for the sub problems LP-relaxation was rounded down to

obtain the bound, because any feasible solution for the sub problem must have an integer Z.Now, with some of the variables not integer restricted, the bound is the value of Z without rounding down.

The fourth (and final) change to the BIP algorithm to obtain our MIP algorithm involves fathoming test 3. Before, with a pure integer Programming problem, the test was that the optimal solution for the sub problem's LP-relaxation is integer. Since this ensures that the solution is feasible and therefore optimal, for the sub problem. Now, with a mixed integer-programming problem, the test requires only that the integer restricted variables be integers in the optimal solution for the sub problem's LP-relaxation, because this suffices to ensure that the solution is feasible, and therefore optimal for the sub problem.

Summary of MIP Branch-And-Bound Algorithm:

BRANCHING: Among the remaining (unfathomed) sub problems, select the one that was created most recently. (Break ties according to which has the larger bound). Among the integer restricted variables that have a non-integer value in the optimal solution for the LP-relaxation of the sub problem, Choose the first one in the natural ordering of the variables to be the branching variable. Let x_j be this variable, and x_j^* its value in the solution. Branch from the node from the sub problem to create two new sub problems by adding the respective constraints, $x_j \leq \lfloor x_j^* \rfloor$ and $x_j \geq \lfloor x_j^* \rfloor + 1$.

BOUNDING: For each new sub problem, obtain its bound by applying the simplex method to its LP-relaxation and using the value of Z for the resulting optimal solution.

FATHOMING: For each new sub problem, apply the three fathoming tests given below, and discard those sub problems that are fathomed by any of the tests.

Test 1: Its bound less than Z^*, where Z^* is the value of Z for the current incumbent.

Test 2: Its LP-relaxation has no feasible solutions.

Test 3: The optimal solution for its LP-relaxation has integer values for the integer-restricted variables (If the solution is better than the incumbent, it becomes the new incumbent and test 1 is reapplied to all unfathomed sub problem with the new larger Z^*.

OPTIMALITY TEST: Stop, when there are no remaining sub problems; the current incumbent is optimal. Otherwise, return to perform another iteration.

3.4 MIXED ZERO-ONE PROBLEM

The mixed zero-one linear problem is defined as:

$$\text{Maximize} \quad z = \sum_{j \in P} c_j x_j + \sum_{j \in Q} d_j y_j$$

$$\text{Subject to} \quad \sum_{j \in P} a_{ij} x_j + \sum_{j \in Q} e_{ij} y_j + S_i = b_i, \qquad i \in M \equiv \{1, \ldots\ldots, m\}$$

$$x_j \geq 0, \qquad j \in P \equiv \{1, \ldots\ldots, p\}$$

$$y_j = (0,1), \qquad j \in Q \equiv \{1, \ldots\ldots, q\}$$

$$S_i \geq 0, \qquad i \in M.$$

There are three special enumeration methods for solving the above problem due to Driebeek [18], (1966), Benders [11], (1962), and Lemke and Spielberg [33], (1967). The Lemke-Spielberg method is closely related to Bender's. A fourth method was developed by R.E. Davis et al. [17], (1971). This method, however, may be regarded as a special case of the Beale-Small algorithm [8], (1965) for the general mixed integer problem.

THE PENALTY ALGORITHM:

Driebeek was the first to conceive the use of penalties in solving mixed integer linear programs. His idea is to first solve the problem as a continuous linear program, that is, with the conditions $y_j = (0,1)$ replaced by $0 \le y_j \le 1$, for all $j \in Q$. In general, some of the y_j variables may be fractional in the optimum continuous solution. If these variables are forced to assume integer values then the objective value of the linear program will decrease. The same result holds if a binary variable y_j at zero (one) level in the continuous solution is forced to assume the value one (zero).

Let z_{max} be the optimum objective value of the linear program and let δ be the decrease from z_{max} resulting from the imposing restrictions on y_j such that a specific binary assignment for all $y_j, j \in Q$, is realized. Then the associated optimum objective value is $z_{max} - \delta$. Now, if z^* is a known lower bound on the optimum objective value of the mixed 0-1 problem, then the specific binary assignment resulting δ may be discarded if $z_{max} - \delta \le z^*$. By enumerating all 2^q binary combinations and updating z^* each time an improved solution is attained, then at the termination z^* and its associated solution give the optimum solution. However, in order for this procedure to reach the sophisticated level of a potentially efficient algorithm, some of the 2^q combinations must be discarded automatically, that is, enumerated implicitly.

In the above outline, one notices that δ can be determined exactly only by solving a linear program for each binary assignment. This may be extremely costly from tl e computational standpoint. What Driebeek proposes is to estimate a lower bound $\underline{\delta}$ on the value of δ, by using the information in the continuous optimum linear program only. Although $z_{max} - \underline{\delta}$ is not as strong as $z_{max} - \delta$, it is evident that a binary assignment yielding $z_{max} - \underline{\delta} \le z^*$ must be discarded. The weakening

of the condition can be tolerated since, as will be shown, $\underline{\delta}$ can be computed easily.

The lower bound $\underline{\delta}$ defines the so-called *penalty*, and the next section shows how penalties are determined. However, if Driebeek's method is followed exactly, one would deal with an enlarged continuous problem having $(m+2q)$ explicit constraints and $(p+3q+m)$ variables in order to compute the penalties. This is actually unnecessary as the simplified version (due to Taha) will show. It must be stressed, however, that the basic steps of the algorithm (after the penalties are computed) are essentially those of the Driebeek.

THE ALGORITHM

Let z^* be the current basic objective value associated with a feasible solution. If no initial basic feasible solution is available, take $z^* = -\infty$. Solve the continuous linear program and record its optimum objective value $z_{\max}$.

Then compute the penalties associated with the binary variables.

Step 0: Select a desired level for each y_j variable. A good initial choice is to select the binary value yielding the lower penalty. Go to step 1.

Step 1: Compute the penalty $\underline{\delta}$ associated with the current binary assignment.

(a) If $z_{\max} - \underline{\delta} \le z^*$, discard the current integer combination as non-promising and go to step 2. Otherwise,

(b) Substitute the binary values of y_j in the mixed integer problem and solve the resulting linear program. If no feasible solution exists or if the objective value drops below z^*, discard the current integer combination and 2^q go to Step 2. Otherwise, the resulting solution yields an improved value of z^*. Record this value and go to step 2.

Step 2: If all 2^q binary combinations have been enumerated, stop; z^*, when finite, gives the optimum. Otherwise generate a new binary combination and go to step 1.

The efficiency of Driebeek's algorithm is dependent in the first place on how effective the penalties are in producing the smallest upper bound, and also on how good initial z^* is. One notices, however, that because the penalties are computed once and for all from the continuous tableau, they have the disadvantage that better values cannot be generated as more binary combinations are enumerated.

CHAPTER IV

NONLINEAR MIXED INTEGER PROGRAMMING

4.1 Introduction:

Nonlinear programming problems exist when linear programming model elements, including the objective function and side constraints, are represented by nonlinear expressions. Like linear mixed integer programming (LMIP) problem, nonlinear mixed integer programming (NLMIP) is a mathematical technique for determining the optimal solutions to many business problems. In LMIP, it is assumed that the unit profit contribution or the cost of production does not change at different levels of production. In actual business operations, however, purely linear relationship may not exist in the profit or cost function when the production activities vary. The average cost of production may change, as the production levels are varied because of the realization of economies or diseconomies of scale, or diminishing in original productivity or the productive factors. As a consequence of these factors, the objective function is nonlinear, or one or more of the side constraint inequalities have nonlinear relationship or both.

The NLMIP can be defined as follows:

Maximize (or minimize) f(**x**)

Subject to $g_j(\mathbf{x}) \leq (\text{or} \geq)\ 0,\quad j = 1,2,\ldots,m.$

x_i integers, $\quad i = 1,2,\ldots,n\ \ (n \leq m)$

where f and g_j are real valued functions.

In the field of nonlinear integer programming Reiter and Rice [37] suggested a method for solving a general quadratic programming problem, where both the objective and constraint functions are quadratic. Another approach, based on the

concept of penalty functions, was suggested by Gellatly and Marcal [20]. This approach was later applied by Gisvold and Moe [25] to solve some design problems, which have been formulated as NLMIP problems. Here we shall discuss the polynomial programming and separable programming approach. The techniques of quadratic programming and branch-and-bound will also be presented.

4.2 Polynomial programming:

Watters [42] has developed a procedure for converting integer polynomial programming problems to zero-one LP problems. The resulting zero-one LP problem can be solved conveniently by the Balas method.

Consider the optimization problem:

Minimize f(**x**)

Subject to $g_j(\mathbf{x}) \leq 0$, j = 1,2,...,m. .....(1)

x_i = integer, i = 1,2,...,n.

where f and g_j, j = 1,2,...,m are polynomials in the variables $x_1, x_2, \ldots, x_n$. A typical term in the polynomials can be represented as

$$c_k \prod_{i=1}^{n_k} (x_i)^{a_{ki}} \qquad \ldots\ldots(2)$$

where c_k is constant, a_{ki} is a nonnegative constant exponent, and n_k is the number of variables appearing in the k^{th} term. We shall convert the integer polynomial programming problem stated in eq. (1) into an equivalent zero-one LP problem in two stages. In the first stage, we shall see how an integer variable, x_i, can be represented by an equivalent system of zero-one (binary) variables. We shall

consider the conversion of a zero-one polynomial programming into a zero-one LP problem in the second stage.

(i) Representation of an integer variable by an equivalent system of binary variables:

Let x_i be an integer variable whose upper bound is given by u_i so that

$$x_i \le u_i \le \infty \qquad \text{.....(3)}$$

We assume that the value of the upper bound u_i can be determined from the constraints of the given problem.

We know that in decimal number system, an integer p is represented as

$$p = p_0 + 10^1 p_1 + 10^2 p_2 + \ldots\ldots, \quad 0 \le p_i \le (10-1=9) \text{ for } i = 0, 1, 2, \ldots\ldots$$

and written as $p = \ldots p_2 p_1 p_0$ by neglecting the zeros to the left. In a similar manner, the integer p can also be represented in the binary number system as

$$p = q_0 + 2^1 q_1 + 2^2 q_2 + 2^3 q_3 \ldots\ldots$$

where $0 \le q_i \le (2-1=1)$ for $i = 0,1,2,\ldots\ldots$

In general, if $y_i^{(0)}, y_i^{(1)}, y_i^{(3)}, \ldots$ denote binary numbers, the variable x_i can be expressed as $x_i = \sum_{k=0}^{N_i} 2^k y_i^{(k)}$ .....(4)

Where N_i is the smallest integer such that

$$\left(\frac{u_i+1}{2}\right) \le 2N_i \qquad \text{.....(5)}$$

Thus the value of N_i can be selected for any integer variable x_i once its upper bound u_i is known.

Method of finding $q_0, q_1, q_2 \ldots$: Let M be the given positive integer. To find its binary representation $q_n q_{n-1} \ldots q_1 q_0$, we compute the following recursively:

$$b_0 = \mathrm{M}$$

$$b_1 = (b_0 - q_0)/2$$

$$b_2 = (b_1 - q_1)/2$$

$$\vdots$$

$$b_k = (b_{k-1} - q_{k-1})/2$$

where q_k=1 if b_k is odd and $q_k = 0$ if b_k is even. The procedure terminates when $b_k = 0$.

(ii) **Conversion of a zero-one polynomial programming problem into a zero-one LP problem:**

The conversion of a polynomial programming problem into a LP problem is based on the fact that

$$x_i^{a_{ki}} \equiv x_i \qquad \ldots\ldots(6)$$

if x_i is binary variable and a_{ki} is a positive exponent. If $a_{ki} = 0$, then obviously the variable x_i will not be present in the k^{th} term. The use of equation (6) permits us to write the k^{th} term of the polynomial, equation (2), as

$$c_k \prod_{i=1}^{n_k} (x_i)^{a_{ki}} = c_k \prod_{i=1}^{n_k} x_i = c_k (x_1 . x_2 \ldots . x_{n_k}) \qquad \ldots\ldots(7)$$

Since each of the variables $x_1, x_2, \ldots$ can take a value of either 0 or1, the product $(x_1 . x_2 . \ldots x_{n_k})$ also will take a value of 0 or 1. Hence by defining a variable y_k as:

$$y_k = x_1.x_2....x_{n_k} = \prod_{i=1}^{n_k} x_i \qquad .....(8)$$

the k^{th} term of the polynomial simply becomes $c_k y_k$. However, we need to add the following constraints to ensure that $y_k = 1$ when all $x_i = 1$ and zero otherwise:

$$y_k \geq \left(\sum_{i=1}^{n_k} x_i\right) - (n_k - 1) \qquad .....(9)$$

$$y_k \leq \frac{1}{n_k}\left(\sum_{i=1}^{n_k} x_i\right) \qquad .....(10)$$

It can be seen that if all $x_i = 1$, $\sum_{i=1}^{n_k} x_i = n_k$, and equations (9) and (10) yield

$$y_k \geq 1 \qquad .....(11)$$

and

$$y_k \leq 1 \qquad .....(12)$$

which can be satisfied only if $y_k = 1$. If at least one $x_i = 0$, we have $\sum_{i=1}^{n_k} x_i < n_k$, and equations (9) and (10) give

$$y_k \geq -(n_k - 1) \qquad .....(13)$$

and

$$y_k < 1 \qquad .....(14)$$

Since n_k is a positive integer, the only way to satisfy equations (13) and (14) under all circumstances is to have $y_k = 0$.

4.3 Separable programming:

An NLP problem is said to be a separable programming (SP) problem if all its function can be separated, each with a single variable. A simple case of separable integer programming has been considered by Fox [19] of the following form:

$$\text{Maximize } z = \sum_{i=1}^{n} f_i x_i = f(\mathbf{x})$$

$$\text{Subject to } g(\mathbf{x}) = \sum_{i=}^{n} g_i(x_i) \leq \mathbf{b}$$

where $g_i, \mathbf{b} > \mathbf{0}$, $x_i \geq \mathbf{0}$ and integer, i = 1,2,...n.

It is assumed that all functions $f_i (i = 1,2,\ldots,n)$ are concave and strictly increasing. This may be applied to the problem of allocating resources or a quadratic knapsack problem.

R.R.Meyer (1977) used the above method for the solution of a very special class of nonlinear integer programs. It was shown that a given nonlinear integer program can be made equivalent to a mixed integer linear program through a piecewise linear approximation to the separable integer programming problem. His problem was composed of a separable objective function and linear constraints. The problem can be stated as:

$$\left.\begin{array}{l} \text{Minimize } z = \sum_{i=1}^{n} f_i(x_i) \\ \text{Subject to } \mathbf{cx} = \mathbf{d} \\ \mathbf{A} \leq \mathbf{x} \leq \mathbf{B} \end{array}\right\} \quad \ldots\ldots(1)$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, $\mathbf{A} = (a_1, a_2, \ldots, a_n)$ and $\mathbf{B} = (b_1, b_2, \ldots, b_n) \in E^n$

and x_i is a nonnegative integer for i = 1,2,...,n.

It is further assumed that c is a totally unimodular matrix and the vector **d** is an integral vector. All the functions $f_i(x_i)$ are assumed to be convex on $[a_i, b_i]$.

Let ϕ_i be the continuous piecewise linear approximation of $f_i(x_i)$ defined on $[a_i, b_i]$, that coincides with $f_i(x_i)$ at the integer points in $[a_i, b_i]$ and is linear between each pair of adjacent integers in this interval. Now consider the problem:

$$\left.\begin{array}{l} \text{Minimize } z = \sum \phi_i(x_i) \\ \text{Subject to } \quad \mathbf{cx} = \mathbf{d} \\ \mathbf{A} \le \mathbf{x} \le \mathbf{B} \end{array}\right\} \qquad \ldots\ldots(2)$$

Where **x**, **A**, **B**, $\in E^n$ and x_i are nonnegative integers.

Since z of (1) and (2) coincide over their common feasible sets (the set of integers), these two problems give equivalent integer solutions.

Let E_i denote the set of integers in $[a_i, b_i]$. Then $\phi_i(x_i)$ has the following form:

$$\phi_i(x_i) = \min \sum_{j \in E_i} \lambda_{ij} f_i(j) \qquad \ldots\ldots(3)$$

where (i) $\sum_{j \in E_i} j.\lambda_{ij} = x_i$

(ii) $\sum_{j \in E_i} \lambda_{ij} = 1$

(iii) $\lambda_{ij} \ge 0$ for i = (1,2,....,n) and $j \in E_i$

No two λ_{ij} are positive unless they are adjacent. Substituting each $\phi_i(x_i)$ in (2) with the help of (3), equation (2) can be represented as:

$$\text{Minimize } \sum_{i=1}^{n} \sum_{j \in L_i} \lambda_{ij} f_i(j) \qquad \ldots\ldots(4)$$

Subject to $\mathbf{cx} = \mathbf{d}$, $\mathbf{x} \geq \mathbf{0}$ and integer,

$$\sum_{j \in L_i} j\lambda_{ij} = x_i \quad \text{for i} = (1,2,\ldots,n),$$

$$\sum_{j \in L_i} \lambda_{ij} = 1 \quad \text{for i} = (1,2,\ldots,n),$$

$$\lambda_{ij} \geq 0$$

and no two λ_{ij} are positive unless they are adjacent (for i = 1,2,....,n).

The representation (4) is a piecewise linear approximation to the separable integer programming problem (1). Again, the formulation (4) is a mixed integer linear program. Therefore, it is shown that a given nonlinear integer program of the type (1) can be equivalent to a mixed integer linear program.

The integer requirements on the vector **x** in (4) may now be removed and hence, the problem reduces to a continuous linear program. Meyer has proved that $(\lambda^*, \mathbf{x}^*)$, an optimal solution to this linear program, will always result in a set of integer solutions for the vector **x**. Thus $(\lambda^*, \mathbf{x}^*)$ automatically becomes an optimal solution to (4), and hence to (1). Therefore, the original nonlinear integer program can be solved by computing the values of each of $f_i(x_i)$ at the integer points in $[a_i, b_i]$ and solving the linear program (4) (removing the integer requirements on x_i), which will generate an optimal extreme point.

4.4 Quadratic programming:

Aggarwal [1] used the cutting plane method introduced by Gomory [24] for solving mixed integer convex quadratic programming problem (QPP).

Consider the following problem:

$$\left.\begin{array}{l} \text{Minimize } Q(x) = \mathbf{p'x} + \mathbf{x'cx} \\ \text{Subject to } \mathbf{Ax} = \mathbf{b} \\ \mathbf{x} \geq \mathbf{0} \end{array}\right\} \quad \ldots\ldots(1)$$

and x_j is an integer for all $j \in J$,

where,

$J = [\, J / x_j$ is an integer],

A is an $m \times n$ matrix,

b is an m component vector,

p' and **x'** are n components row vectors in R^n,

c is a symmetric matrix of order $n \times n$,

x'cx represents a quadratic form which is said to be positive semi definite or negative semi definite according as $\mathbf{x'cx} \geq 0$ or < 0, respectively, for $\mathbf{x} \neq \mathbf{0}$.

Assume that **x'cx** is a positive semi definite, that is, $Q(\mathbf{x})$ is convex. It is also assumed that the constraints are feasible, the feasible set is bounded and degeneracy is absent.

The Algorithm:

The problem (1) is first solved by Beal's method. Let us call $x_1, x_2, \ldots, x_n$ as proper variables. During the application of Beale's method all the

free variables, which have been made basic at any stage, should not be considered further. During variations of non-basic variables the equations of the proper variables are used to keep these basic free variables non-negative. Because, in order to introduce Gomory type cuts all the variables, proper and free, should be non-negative. We will now call a free variable as improper variable as it is no more free.

Let **x** denote the optimum solution to eq. (1).

Then we must have,

$$\frac{\delta Q}{\delta x_j} \geq 0, \text{ for all non-basic } x_j \text{ and } \frac{\delta Q}{\delta U_k} = 0, \text{ for all non-basic } U_k.$$

Where, U_k,- $k = 1,2,\ldots,n$ are free variables at the final test point and the objective function Q(**x**) is in terms of only non-basic variables.

If all x_j, j∈J are integers, **x** will be the optimum solution to equation (1).

Let all x_j, j∈J are not integers. Select any of these x_j'' say x_p, then x_p can be expressed as

$$x_p = a_{p0} + a_{pj}(-x_j) + a_{pk}(-U_k) \qquad \ldots\ldots(3)$$

where the two summations are for all non-basic proper variables and non-basic improper variables. Clearly, a_{p0} is non-integral.

Let us denote the integral and fractional parts of a_{p0}, a_{pj} and a_{pk} by $[a_{p0}], Q_{p0}$; $[a_{pj}], Q_{pj}$; $[a_{pk}], Q_{pk}$ respectively.

The Gomory cut can now be introduced as a basic variable S.

Where,

$$S = -Q_{p0} + \sum(-Q_{pj})(-x_j) + \sum(-Q_{pk})(-U_k) + \sum_{[j=j,a_{pj}<0]}(-a_{pj})(-x_j) + \sum_{[j+j,a_{pj}<0]}(Q_{p0}a_{pj}/1-($$

$$+ \sum_{[k=J,a_{pk}<0]}(-a_{pk})(-U_k) + \sum_{[k+J,a_{pk}<0]}(Q_{p0}a_{pk}/1-Q_{p0}).(-U_k).....(4)$$

The problem (1) can now be solved with (4) as an additional constraint by parameter 't' method introduced by Beale [9].

Define $S_t = S + t$ .....(5)

where t is the Beal's parameter. Clearly the value of t for which the present solution is feasible is Q_{p0}.

The parameter t method now gradually decreases the value of t to zero. If $t < Q_{p0}$ then $S_t \leq 0$ and S_t will become non-basic. If S_t contains any non-zero term in any improper variable, this should be made basic. If such improper variable is not unique, any one of them could be chosen first. The process terminates when t = 0 without any basic proper variables or any partial derivative of Q becoming negative. If at this stage we still have some non-integer values of the variables, which are constrained to be integers, more cuts are added one by one and the process is repeated until we reach the required optimal solution.

Numerical example based on mixed integer QPP:

Consider the following problem

$$\left.\begin{array}{l} \text{Minimize } Q(\mathbf{x}) = 183\text{-}44x_1 - 42x_2 + 8x_1^2 - 12x_1x_2 + 17x_2^2 \\ \text{Subject to } 2x_1 + x_2 \leq 10 \\ \qquad x_1, x_2 \geq 0 \\ \text{and} \qquad x_1 \text{ is an integer.} \end{array}\right\} \quad .....(1)$$

Introdu :ing slack variable $x_3 \geq 0$, constraints can be written as

$$2x_1 + x_2 + x_3 = 10$$

$$x_1, x_2, x_3 \geq 0$$

The solution to the above problem by Beal's method yield:

$$x_1 = 19/5 + (1/5)U_2 - 2/5x_3.$$

$$x_2 = 12/5 - (2/5)U_2 - (1/5)x_3.$$

$$Q = 19 + 6x_3 + 3x_3^2 + 3x_3^2 + 4U_2^2$$

That is. $x_1 = 19/5$.

$$x_2 = 12/5.$$

and Q(**x**) = 19.

Since x_1 is required to be integer, we thus have the Gomory's cut as

$$S = 4/5 + \frac{(4/5)X_3 - 1/5}{1 - 4/5}(-u_2) + (2/5)X_3$$

Or, S = -4/5 + $(4/5)U_2 + (2/5)X_3$ .....(2)

Addition of parameter t to (2) gives:

$$S_1 = -4/5 + t + (4/5)U_2 + 2/5x_3$$

S_1 will now become non-basic in place of U_2,

we have,

$$U_2 = 1 - (5/4)t + (5/4)S_1 - (1/2)x_3$$

$$x_1 = 4 - (1/4)t + (1/4)S_1 - (1/2)x_3$$

$$x_2 = 2 + (1/2)t + (1/2)S_1$$

and

$$Q = 19 + 6x_3 + 3x_3^2 + 4(1 - (5/4)t + (5/4)S - (1/2)x_3)^2$$

T is now reduced to zero without making x_1 and x_2 or any partial derivatives of Q negative, which yields:

$$x_1 = 4,$$
$$x_2 = 2,$$

$$Q = 23$$

This will be the required solution to the mixed integer QPP given by (1).

4.5 Branch-and-bound methods:

It has long been accepted that the branch-and-bound principle is an effective computational tool for solving mixed integer linear programming problems. In this section we investigate the computational feasibility of branch-and-bound methods in solving nonlinear mixed integer programming problems. We have discussed in chapter II that a branch-and-bound method becomes an efficient device when the rules used for selecting the branching variables and branching node are carefully and properly adopted. The branch-and-bound strategy for NLMIP or NLIP problems follows in the same lines as that for MILP.

The techniques discussed in the chapter II can readily be extended to the nonlinear problem excepting those properties based on linearity assumption. These typically include the use of penalties in estimating the bounds on the objective value and the development of certain branching rules.

Hence, the original Land and Doig algorithm is not suitable, in general, to solve the nonlinear integer program, primarily because the validity of the branching rule is tied with the assumption of linearity. However, Dakin's

modification of the Land and Doig algorithm makes the branching rule independent of the linearity condition. Suppose

$x_j = x_j^0$ is the optimum value of x_j, which is noninteger, then for x_j to assume an integer value at the optimum it must satisfy either of the two conditions: $x_j \geq [x_j^0]$ and $x_j \leq [x_j^0]$. This is free from linearity restriction.

The major difficulty with the solution of a nonlinear program lies generally in finding a solution method that guarantees the determination of the global optimum at each node. For instance, if such methods can find local optima only, then the branch-and-bound principle becomes inapplicable since the bounding rule cannot be satisfied. It is evident that the said complication does not arise as a result of using branch-and-bound principle. Rather, the techniques of nonlinear programming still have limited capabilities for solving the integer relaxed problems. It has, however, been shown by Kunzi and Dettli [29], Agarwal and Swaroop [2], Bari and Arshad [5] etc., that good results are obtained with nonlinear algorithms if the constraint and the objective function of the problem satisfy certain conditions of convexity and concavity so that a local optimum becomes the global one.

CHAPTER V

PARTITIONING IN MIXED INTEGER PROGRAMMING

5.1 Introduction:

Mixed integer programming can also be solved using partitioning. The details of the algorithm are given below:

Consider a mixed integer program

$$\text{Minimize } z = \mathbf{c}_1\mathbf{x} + \mathbf{c}_2\mathbf{y}$$

$$\text{Subject to } \mathbf{A}_1\mathbf{x} + \mathbf{A}_2\mathbf{y} \geq \mathbf{b} \qquad \ldots\ldots(1)$$

$$\mathbf{x}, \mathbf{y} \geq \mathbf{0}, \quad \mathbf{y} \equiv \mathbf{0} \pmod 1,$$

i.e., only **y** is restricted to be an integer vector.

For given values of **y**, problem (1) becomes

$$\text{Minimize } \mathbf{c}_1\mathbf{x} \qquad \ldots\ldots(2)$$

$$\text{Subject to } \mathbf{A}_1\mathbf{x} \geq \mathbf{b} - \mathbf{A}_2\mathbf{y}, \quad \mathbf{x} \geq 0$$

Then (2) is a standard linear program. The dual problem of (2) is

$$\text{Maximize } \mathbf{u}(\mathbf{b} - \mathbf{A}_2\mathbf{y})$$

$$\text{Subject to } \mathbf{u}\mathbf{A}_1 \leq c_1, \quad \mathbf{u} \geq \mathbf{0} \qquad \ldots\ldots(3)$$

We note two interesting features of (3). First, the feasible region of **u** defined by $\mathbf{u}\mathbf{A}_1 \leq \mathbf{c}_1$, is independent of y. Second, no matter what values y may take, the maximum of $\mathbf{u}(\mathbf{b}-\mathbf{A}_2\mathbf{y})$ always occurs on a vertex of the convex polytope defined by $\mathbf{u}\mathbf{A}_1 \leq \mathbf{c}_1$, provided that the convex polytope is bounded from above. Let $\mathbf{u}^p$ (p = 1,2,...,P) denote a generic vertex of the convex polytope. Then we may write (3) as

$$\text{Maximize } \mathbf{u}^p(\mathbf{b} - \mathbf{A}_1\mathbf{y})$$

$$\text{Subject to } \mathbf{u}^p \geq \mathbf{0} \quad (p = 1,2,\ldots,P) \qquad \ldots\ldots(3')$$

If (3) has no feasible solution, then from duality theory, (2) either has no finite optimum solution or also has no feasible solution. If (3) has no finite optimum solution, then (2) has no feasible solution. Either case would apply that (1) has no finite optimum solution. Thus, we are interested only in the case that (3) has a finite optimum solution. This means that the convex set $\mathbf{u}\,\mathbf{A}_1 \leq \mathbf{c}_1$ is not empty, but it does not imply that the convex set $\mathbf{u}\,\mathbf{A}_1 \leq \mathbf{c}_1$ is bounded. It is possible for the convex set $\mathbf{u}\,\mathbf{A}_1 \leq \mathbf{c}_1$ to be unbounded and for $\mathbf{u}$ to go to infinity for certain values of $(\mathbf{b} - \mathbf{A}_2\mathbf{y})$ and yet the optimum vertex associated with the optimum value of y to still have finite coordinates. The situation is shown in figure 1, where the 'o' denotes the optimum vertex and the convex set is unbounded.

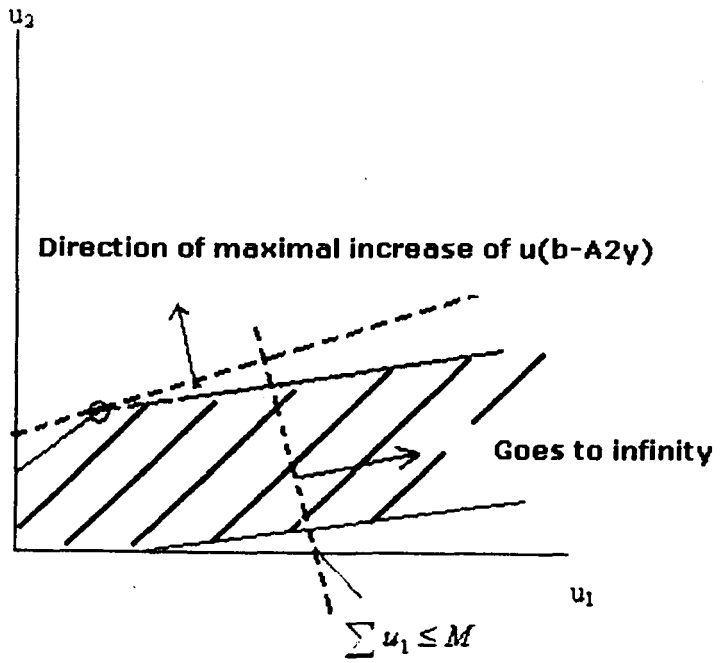


Figure 1

In the case in which u goes to infinity for certain values of $(\mathbf{b} - \mathbf{A}_2\mathbf{y})$ we can add the constraints $\sum u_i \le M$ (where M is a very large positive constant) to the constraint set $\mathbf{u}\mathbf{A}_1 \le \mathbf{c}_1$. this is shown by the long dashed line in figure 1.

If the convex set $\mathbf{u}\mathbf{A}_1 \le \mathbf{c}_1$ is bounded from above or if it becomes bounded from above after the constraint $\sum u_i \le M$ is added, then (3) and (3') are equivalent programs, and this is the case we are interested in. We do not know whether $\mathbf{u}\mathbf{A}_1 \le \mathbf{c}_1$ is bounded or not before solving the program (3). Let us substitute (3,) into (1) and we have

$$\text{Minimize } z$$

$$\text{Subject to } z \ge \mathbf{c}_2\mathbf{y} + \max_{u^p} \mathbf{u}^p(\mathbf{b} - \mathbf{A}_2\mathbf{y}), \qquad \ldots\ldots(1')$$

$$\mathbf{u}^p \ge \mathbf{0} \quad (p = 1,2,\ldots,P)$$

Each $\mathbf{u}^p$ gives one constraint and (1') is a pure integer program for a fixed number of $\mathbf{u}^p$.

If we know the optimum vertex $\mathbf{u}^*$, we can solve (1') as a pure integer program by any of the existing methods to obtain optimum values z^* and $\mathbf{y}^*$. Substituting the optimum $\mathbf{y}^*$ into (2), we solve the linear program min $\mathbf{c}_1\mathbf{x}$ subject to $\mathbf{A}_1\mathbf{x} \ge \mathbf{b}-\mathbf{A}_2\mathbf{y}^*$ and obtain the optimum value $\mathbf{x}^*$. Substituting $\mathbf{x}^*$ and $\mathbf{y}^*$ into (1), we have $z = \mathbf{c}_1\mathbf{x}^*+\mathbf{c}_2\mathbf{y}^*$. This value z should, of course, be the same as the z^* obtained from (1'). If we do not know the optimum vertex, we can list all vertices $\mathbf{u}_p$ and solve (1'). As there are too many vertices $\mathbf{u}^p$, so we shall use the following iterative procedure to get the optimum vertex $\mathbf{u}^*$. Take any feasible $\bar{\mathbf{u}}$ (not necessarily a vertex) that satisfies $\bar{\mathbf{u}}\mathbf{A} \le \mathbf{c}_1$, substitute it into (1'), and let the solution to (1') for this value of $\bar{u}$ be $\bar{z}$ and $\bar{\mathbf{y}}$. Use $\bar{\mathbf{y}}$ in (3) and solve for $\mathbf{u}$ that maximizes $\mathbf{u}(\mathbf{b}-\mathbf{A}_2\bar{\mathbf{y}})$. Let the solution be $\bar{\bar{\mathbf{u}}}$. Put $\bar{\bar{\mathbf{u}}}$ into (1'), i.e., add one or more inequality and solve for $\mathbf{y}$ again; this is iterated until the optimum $\mathbf{u}^*$ is obtained.

For optimum $\mathbf{u}^*$, $\mathbf{y}^*$, and z^* we have from (1)

$$z^* = \mathbf{c}_2\mathbf{y}^* + \mathbf{u}^*(\mathbf{b} - \mathbf{A}_2\mathbf{y}^*).$$

For a given $\bar{\mathbf{u}}$, we have from (1')

$$\bar{z} = \mathbf{c}_2\bar{\mathbf{y}} + \bar{\mathbf{u}}(\mathbf{b} - \mathbf{A}_2\bar{\mathbf{y}}),$$

and $\bar{z} < z^*$ since $\bar{z} \geq \mathbf{c}_2\bar{\mathbf{y}} + \bar{\mathbf{u}}(\mathbf{b} - \mathbf{A}_2\bar{\mathbf{y}})$ is only a subset of all the constants in (1'). The true optimum z^* is obtained in (1') only when all $\mathbf{u}^p$ are used or optimum $\mathbf{u}^*$ is used. To check if a vertex $\bar{\mathbf{u}}$ is an optimum vertex, we first use $\bar{\mathbf{u}}$ to solve (1') and get $\bar{\mathbf{y}}$ and $\bar{z}$ and then use $\bar{\mathbf{y}}$ in (3) to maximize $\mathbf{u}(\mathbf{b}-\mathbf{A}_2\bar{\mathbf{y}})$. If $\bar{z} - \mathbf{c}_2\bar{\mathbf{y}} = \max_{u} \mathbf{u}(\mathbf{b} - \mathbf{A}_2\bar{\mathbf{y}})$, then $\bar{\mathbf{u}}$ is the optimum vertex.

5.2 The Algorithm:

Step 1- Start with a $\bar{\mathbf{u}} \geq \mathbf{0}$ that satisfies $\mathbf{u}\mathbf{A}_1 \leq \mathbf{c}_1$. This $\bar{\mathbf{u}}$ does not have to be a vertex. If none exists, then the original problem (1) has no feasible solution. Go to step 2.

Step 2- solve the pure integer program

Minimize z

Subject to $z \geq \mathbf{c}_2\mathbf{y} + \bar{\mathbf{u}}(\mathbf{b} - \mathbf{A}_2\mathbf{y})$

$\mathbf{y} \geq \mathbf{0}, \quad \mathbf{y} \equiv \mathbf{0} \pmod 1.$

If z is unbounded from below, take a z to be any small value $\bar{z}$.

Step 3- Using $\bar{\mathbf{y}}$ obtained in step 2, we solve the linear program

$$\max_{u} \mathbf{u}(\mathbf{b} - \mathbf{A}_2\bar{\mathbf{y}})$$

subject to $\mathbf{u}\mathbf{A}_1 \leq \mathbf{c}_1, \ \mathbf{u} \geq \mathbf{0}.$

If $\mathbf{u}$ goes to infinity with $\mathbf{u}(\mathbf{b} - \mathbf{A}_2\bar{\mathbf{y}})$ finite, add the constraints $\sum u_i \leq M$, where M is a large positive constant, and resolve this problem.

Let the solution of this problem be $\bar{\bar{\mathbf{u}}}$. Determine whether

$$\bar{z} - \mathbf{c}_2\bar{\mathbf{y}} \leq \bar{\bar{\mathbf{u}}}\,(\mathbf{b} - \mathbf{A}_2\bar{\mathbf{y}}).$$

If the inequality is satisfied, go to step 4. If it is not satisfied, go to step 2, and add $z \geq \mathbf{c}_2\mathbf{y} + \bar{\bar{\mathbf{u}}}(\mathbf{b} - \mathbf{A}_1\bar{\mathbf{y}})$ to the existing set of constraints in (1').

Step 4- Use $\bar{\mathbf{y}}$ obtained in step 2. Solve the linear program

Minimize $\mathbf{c}_1\mathbf{x}$

Subject to $\mathbf{A}_1\mathbf{x} \geq \mathbf{b} - \mathbf{A}_2\bar{\mathbf{y}}$, $\mathbf{x} \geq \mathbf{0}$.

Let the solution be $\bar{\mathbf{x}}$.. We claim that $\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$ are then the optimal solution and $\mathbf{z}^* = \mathbf{c}_1\bar{\mathbf{x}} + \mathbf{c}_2\bar{\mathbf{y}}$.

5.3 Numerical Illustration:

Consider the following example:

Minimize $5x + 2y_1 + 2y_2 \geq 5$

Subject to

$$x + 3y_1 + 2y_2 \geq 5$$
$$4x - y_1 + y_2 \geq 7 \qquad \ldots\ldots(1)$$
$$2x + y_1 - y_2 \geq 4$$

$$x \geq 0, y_1, y_2 \geq 0, y_1 \equiv y_2 \equiv 0(mod\,1)$$

Rewriting (1), we have

Minimize 5x

Subject to

$$\begin{bmatrix} 1 \\ 4 \\ 2 \end{bmatrix} x \geq \begin{bmatrix} 5 - 3y_1 - 2y_1 \\ 7 + y_1 - y_2 \\ 4 - y_1 + y_2 \end{bmatrix}, x \geq 0 \qquad \ldots\ldots(2)$$

The dual program of (2) is

$$\text{Max}(u_1,u_2,u_3)\begin{bmatrix}5-3y_1-2y_2\\7+y_1-y_2\\4-y_1+y_2\end{bmatrix}$$

Subject to

$$\begin{aligned}&u_1+4u_2+2u_3\le 5\\&u_1,u_2,u_3\ge 0\end{aligned} \qquad \ldots\ldots(3)$$

One feasible solution of (3) is $u_1=0, u_2=5/4, u_3=0$.

Rewriting (1) we have

Minimize z

Subject to

$$z\ge 2y_1+2y_2+\max \mathbf{u}^{p}\begin{bmatrix}5-3y_1-2y_2\\7+y_1-y_2\\4-y_1+y_2\end{bmatrix} \qquad \ldots\ldots(1')$$

$$u_1,u_2,u_3\ge 0.$$

Substituting (0,5/4,0) of (3) into (1'), we have

Minimize z

Subject to

$$\begin{aligned}&z\ge 2y_1+2y_2+\tfrac{5}{4}(7+y_1-y_2),\\&y_1,y_2\ge 0\end{aligned} \qquad \ldots\ldots(1'')$$

The solution of (1'') is $\bar{z}=35/4$, and $\bar{y}_1=\bar{y}_2=0$.

Substituting $\bar{y}_1=\bar{y}_2=0$ into (3), we have

Maximize $5u_1+7u_2+4u_3$

Subject to

$$\begin{aligned}&u_1+4u_2+2u_3\le 5,\\&u_1,u_2,u_3\ge 0.\end{aligned} \qquad \ldots\ldots(3')$$

The solution is (5,0,0), with $5u_1+7u_2+4u_3=25$.

Since (35/4) – (2.2)[0 0] <25, we add (5,0,0) into (1'):

Minimize z

Subject to

$$z \geq 2y_1 + 2y_2 + (5/4)(7 + y_1 - y_2),$$
$$z \geq 2y_1 + 2y_2 + 5(5 - 3y_1 - 2y_2),$$
$$y1, y2 \geq 0.$$

The solution is $y_1 = 1$ and $y_2 = 0$, and z = 12.

Substituting $y_1 = 1$ and $y_2 = 0$ into (3), we have

Maximize $2u_1 + 8u_2 + 3u_3$

Subject to $u_1 + 4u_2 + 2u_3 \leq 5,$

$$u_1, u_2, u_3 \geq 0.$$

The solution is $u_1 = 5, u_2 = 0, u_3 = 0,$ with $2u_1 + 8u_2 + 3u_3 = 10.$

Since 10 = z – (2,2)y = 12 – 2×1 - 2×0 = 10, the solution is optimum. Substituting $y_1 = 1, y_2 = 0$ into (2), we have

Minimize 5x

Subject to

$$\begin{bmatrix} 1 \\ 4 \\ 2 \end{bmatrix} x \geq \begin{bmatrix} 2 \\ 8 \\ 3 \end{bmatrix}, \quad x \geq 0.$$

The solution is x = 2 with 5x = 5×2 = 10 as expected. The optimum solution to (8) is x = 2, $y_1 = 1, y_2 = 0,$ and z* = 12.

REFERENCES

1. Agarwal,S.C.,(1974), "On Mixed integer Quadratic Programs",Naval Research Logistics Quaterly. 21, 289-297.

2. Agarwal.S.C. and Swarup.K.(1974), "On Integer Solutions to Quadratic Programs".IV Annual Convention of the Operations Research Society of India, IIT Madras.

3. Agin,N. (1966), "Optimum Seeking with Branch-and-Bound" Management Science, 138, 176-185.

4. Austin,L.M.. and Ghandforoush,P. (1983), "An Advanced Dual Algorithm with Constraint Relaxation for All-Integer Programming", Naval Research Logistics Quarterly. 30,1, 133-144.

5. Bari and Arshad (1979), "A Branch-and-Bound Method for Integer Quadratic Programming". Pure and Applied Mathematika Sciences.

6 Bari and Qazi Shoeb (2003), "NAZ cut for Integer Programming". PAMS, LVII, 87-94.

7. Balas,E. (1968), "A Note on Branch-and-Bound Principle", Operations Research. 16,442-445

8. Beale,E.M.L., and Small,R.E.(1965), "Mixed Integer Programming by Branch-and-Bound Technique". Proc. IFIP.congr., New york May 1965,2,450-451.

9. Beale, E., "A Method of Solving Linear Programming Problems when some but Not All of the Variables Must Take Integral Values", Statistical Techniques Research Group, Technical Report No. 19, Princeton University, July 1958.

10. Bell,D.E. (1979), "Efficient Group Cuts for Integer Programs", Mathematical Programming, 17,2,176-183.

11. Benders.J.F.(1962), "Partitioning Procedures for Solving Mixed-Variables Programming Problems". Numer. Math. 4,239-252

12. Charnes, A.; Granot, d.; Granot, F. (1977), "On Intersection Cuts in Interval Integer Linear Programming", Operations Research, 25, 2, 352-355.

13. Claudia H. Pragman (1993), "An Integer Program for Decision Support in the Charitable Disposition of Excess Inventory". IJOMAS 9, 3, 229-240.

14. Crowder.H,E.L Johnson. and M. Padbergs (1983), "Solving Large Scale Zero-One Linear Programming Problems", Operations Research, 31, 803-834.

15. Dakin.R.J. (1965), "A Tree Search Algorithm for Mixed Integer Programming Problems". Computer Journal. 8. 250-255.

16. Dantzig,G., D. Fulkerson, and S Johnson(1954), "Solution of a Large Scale Travelling Sales- man problem", Operations Research 2(4), 393-410.

17. Davis,R.E., Kendrick,D.A, and Weitzman,M (1971). "A Branch-and-Bound Algorithm for 0-1 Mixed Integer Programming Problem". Operations Research 13. 482-494.

18. Driebeek,N.J (1966), "An Algorithm for the Solution of Mixed Integer Programming Problems". Management Science 12, 576-587.

19. Fox,B. (1966), "Discrete Optimization via Marginal Analysis". Management Science, 13,3,210-216.

20. Gellatly. R.A., and P.B. Marcal, (1967), "Investigation of Advanced Spacecraft Structural Design Technology". NASA report, No. 2356-950001.

21. Ghandforoush,P. and Austin,L.M. (1981), "A Primal-Dual Cutting-Plane Algorithm for All - Integer Programming Algorithm", Naval Research Logistics Quarterly, 28, 4, 559-566.

22. Gomory,R., "An algorithm for Integer Solutions to Linear Programs", in Recent Advances in Mathematical Programming", New York Mc.Graw Hill.

23._____"All Integer Programming Algorithm". IBM Research Center Report RC-189, January 1960; also in Industrial Scheduling, Eanglewood Cliff, NJ, Prentice-Hall 1963.

24. Gomory,R, "Essentials of an Algorithm for Integer Solution to Linear Programs". Bulletin American Mathematical Society,64, 275-278.

25. Gisvold,K.M. and Moe,J. (1972). "A Method for Nonlinear Mixed Integer Programming and its Application to Design Problems". Journal of Engineering for Industry, 352-364.

26. Glover, "A New Foundation for a Simplified Primal Integer Programming Algorithm". Operations Research 13(6), 879-929.

27. Glover, F.; and Sommer,D.; (1972), "Pitfalls of Rounding in Discrete Management Decision Problems", Management Sci. Rep.Ser. No. 72-2. Univ. of Colorado, Boulder.

28. Jeroslow, R. (1979), "The Theory of Cutting Planes", Chapter 2 of Combinatorial Optimization by Christofides, N.; Mingozzi,A.; Toth, P. and Sandi.C. (Ed.), John Wiley and Sons, Ltd., New York.

29. Kunzi,H.P. and Oettli,W.(1963), "Integer Quadratic Programming". IN RECENT ADVANCES IN MATHEMATICAL PROGRAMMING, by Graves,R.L and Welte,P.,303-308, Mc. Graw Hill, New york.

30. Kwak, N.K. (1973), Mathematical Programming with Business Applications, Mc.Graw-Hill, New York.

31. Land,A., and Doig, A.G.(1960), "An Aumatic Method of Solving Discrete Programming Problems", Econometrica, 28, 3, 497-520.

32. Lawler, E.L. and D.E.Wood (1970), "Branch-and-Bound Methods: A Survey",Operations Research , 18, 24-34.

33. Lemke,C.E., and Spielberg, K(1967). "Direct Search Zero-One and Mixed Integer Programming". Operations Research. 15, 892-914.

34. Markowitz,M., and A.Manne (1957), "On the Solution of Discrete Programming Problems". Econometrica 25(1), 84-110.

35. Marsten,R.E, and Muller,M.R. (1980), "A Mixed Integer Programming Approach to Air Cargo Fleet Planning". Management Science. 26,11; 1096-1107.

36. Nemhauser, George L., and Lawrence A. Wolsey (1988), "Integer and Combinatorial Optimization, Wiley, New York.

37. Reiter,S, and D.B. Rice (1966), "Discrete Optimizing Solution Procedures for Linear and nonlinear Integer Programming Problems", Management Science 12, 829-850.

38. Salkin,H.(1971), "A Note on Gomory Fractional Cuts", Operations Research, 19, 6, 1538-1541.

39. Sherali , H.D.(1981), "Expediments for Solving Some Specially Structured Mixed Integer Programs", Nav. Res. Log. Quar., 28,3, 447-462.

40. Taha, Hamdy A, (1975), "Integer Programming Theory, Applications and Computations". Academic Press, New York.

41.——— ; "An Algorithm for the Mixed Integer Program". RM-2597 Rand Corporation, July 1960.

42. Watters, L. (1967), "Reduction of Integer Polynomial Programming Problems to Zero-One Linear Programming Problems", Operations Research, 15, 1171-1173.

43. Young,R (1971)., "A Simplified Primal (All-Integer) Integer Programming Algorithm", Operations Research 16(4), 750-782.